Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ

(Technical Report)

July 4, 2016

Ben Smyth
Huawei Technologies Co. Ltd.,
Boulogne-Billancourt, France
research@bensmyth.com

Steven Frink
Cornell University
Ithaca, NY, US
sfrink@cs.cornell.edu

Michael R. Clarkson
Cornell University
Ithaca, NY, US
clarkson@cs.cornell.edu

***Abstract*—Election verifiability is defined in the computational model of cryptography. The definition formalizes notions of voters verifying their own votes, auditors verifying the tally of votes, and auditors verifying that only eligible voters vote. The Helios (Adida et al., 2009), Helios-C (Cortier et al., 2014) and JCJ (Juels et al., 2010) election schemes are analyzed using the definition. Neither Helios nor Helios-C satisfy the definition because they fail to ensure that recorded ballots are tallied in certain cases when the adversary posts malicious material on the bulletin board. A variant of Helios is proposed and shown to satisfy the definition. JCJ does not satisfy the definition because of a trust assumption it makes, but it does satisfy a weakened definition. Two previous definitions of verifiability (Juels et al., 2010; Cortier et al., 2014) are shown to permit election schemes vulnerable to attacks, whereas the new definition prohibits those schemes.**

I. Introduction

Electronic voting systems that have been deployed in real-world, large-scale public elections place extensive trust in software and hardware. Unfortunately, instead of being trustworthy, many systems are vulnerable to attacks that could bring election outcomes into disrepute [25], [63], [80], [121]. So relying solely on trust in voting systems is unwise; verification of election outcomes is essential.[1]

Election verifiability enables voters and auditors to ascertain the correctness of election outcomes, regardless of whether the software and hardware of the voting system are trustworthy [1], [2], [33], [81], [104]. Kremer et al. [89] decompose election verifiability into three aspects:[2]

- *Individual verifiability:* voters can check that their own ballots are recorded.
- *Universal verifiability:* anyone can check that the tally of recorded ballots is computed properly.
- *Eligibility verifiability:* anyone can check that each tallied vote was cast by an authorized voter.

We propose new definitions of these three aspects of verifiability in the computational model of cryptography. We show that individual and universal verifiability are orthogonal, and that eligibility verifiability implies individual verifiability. Because some electronic voting systems implement voter authentication themselves, whereas other systems outsource voter authentication to third parties, we develop two variants of our definitions—one for systems with *internal authentication* and another for systems with *external authentication*.

We employ our definitions to analyze the verifiability of two well-known election schemes, JCJ [83] and Helios [5]. JCJ is an election scheme that achieves *coercion resistance* and has been implemented as Civitas [37]; it implements its own internal authentication. Helios is a web-based voting system that has been deployed in the real-world and outsources authentication. We also analyze the verifiability of Helios-C [41], a variant of Helios that implements internal authentication by digitally signing ballots.

The Helios 2.0 election scheme is known to have vulnerabilities that can be exploited to violate ballot secrecy and verifiability [21], [44], [45], and the specification for the next Helios release [4], henceforth *Helios'12*, is intended to mitigate against those vulnerabilities. Our analysis shows that the mitigations are insufficient to ensure verifiability. In particular, an adversary could record a ballot that causes a voter's ballot to be omitted from tallying. A variant of Helios, henceforth *Helios'16*, is proposed, and shown to satisfy our definition of election verifiability with external authentication. Helios 2.0 and Helios'12 fail to satisfy our definition.

Our analysis of Helios-C reveals that an adversary could record an ill-formed ballot that causes tallying to abort in a manner that anyone will accept. Yet, our definition of universal verifiability demands that accepted outcomes include the choices used to construct any well-formed ballots. Hence, each voter can be assured that their choice contributed to

1. *Doveryai, no proveryai* (trust, but verify) says the Russian proverb.

2. This decomposition has been criticized [93]; we refute that criticism in Section VIII.

the outcome. By comparison, Helios-C does not assure this, because ill-formed ballots cause tallying to abort and that abort will be accepted. Thus, Helios-C does not satisfy our definition of universal verifiability. Nevertheless, a straightforward variant of Helios-C that disregards ill-formed ballots would satisfy our definition.

The JCJ election scheme does not satisfy our definition of eligibility verifiability, because an adversary who learns the tallier's private key could cast unauthorized votes. We introduce a weakened definition of eligibility verifiability, incorporating JCJ's trust assumption that the private key is not known to the adversary, and show that JCJ satisfies our weakened definition of election verifiability with internal authentication.

Our definitions of election verifiability improve upon two previous definitions [41], [83] by detecting a new class of *collusion attacks*, in which the tallying algorithm announces an incorrect tally, and the verification algorithm colludes with the tallying algorithm to accept the incorrect tally. Examples of collusion attacks include vote stuffing, and announcing tallies that are independent of the election. Our definitions also improve upon those previous definitions by detecting a new class of *biasing attacks*, in which the verification algorithm rejects some legitimate election outcomes. Examples of biasing attacks include rejecting outcomes in which a particular candidate does not win, and rejecting all election outcomes, even correct outcomes.

This paper thus contributes to the security of electronic voting systems by:

- proposing computational definitions of election verifiability;
- showing that individual, universal, and eligibility verifiability are mostly orthogonal properties of voting systems;
- proving that Helios 2.0, Helios'12 and Helios-C do not satisfy election verifiability, and that Helios'16 and JCJ do; and
- identifying collusion and biasing attacks as new classes of attacks on voting systems and demonstrating that they are not detected by two earlier definitions.

Our definitions are sufficient to analyze Helios 2.0, Helios'12, Helios'16, Helios-C, and JCJ. They correctly indentify Helios 2.0, Helios'12, and Helios-C as not satisfying verifiability. And they enable the first proofs that Helios'16 and JCJ satisfy a computational definition of verifiability. Although some protocols may fall outside the scope of our definitions, we have shown that they are sufficiently general to be useful.

Structure: Section II defines election verifiability with external authentication. Section III analyzes Helios. Section IV defines election verifiability with internal authentication. Section V analyzes Helios-C. Section VI analyzes JCJ. Section VII introduces collusion and biasing attacks. Section VIII reviews related work and Section IX concludes. Appendix A defines cryptographic primitives. The remaining appendices explore alternative definitions of verifiability, give the details of Helios and JCJ, and present proofs.

II. External Authentication

Some election schemes do not implement authentication themselves, but instead rely on an external authentication mechanism. Helios, for example, supports authentication with Facebook, Google and Yahoo credentials.[3] In essence, the election scheme outsources ballot authentication. We begin by defining election verifiability for that model.

A. Election scheme

An *election scheme with external authentication*, which henceforth in this section we abbreviate as "election scheme," is a tuple $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ of probabilistic polynomial-time (PPT) algorithms:

- **Setup**, denoted[4] $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$, is executed by the *tallier*, who is responsible for tallying ballots.[5] Setup takes a security parameter k as input and outputs a key pair $(PK_{\mathcal{T}}, SK_{\mathcal{T}})$, a maximum number of ballots m_B, and a maximum number of candidates m_C.[6]
- **Vote**, denoted $b \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$, is executed by voters. A voter makes a *choice* of candidate from a sequence $c_1, \ldots, c_{n_C}$ of candidates. A *well-formed* choice is an integer β, such that $1 \leq \beta \leq n_C$. Vote takes as input the public key $PK_{\mathcal{T}}$ of the tallier, the number n_C of candidates, the voter's choice β of candidate, and security parameter k. It outputs a ballot b, or error symbol $\perp$. An error might occur if the candidate choice is not well-formed or for other reasons particular to the election scheme.
- **Tally**, denoted $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, k)$, is executed by the tallier. It involves a public *bulletin board* BB, which we model as a set.[7] Tally takes as input the private key $SK_{\mathcal{T}}$ of the tallier, the bulletin board BB, the number of candidates n_C, and security parameter k. It outputs a tally $\mathbf{X}$ and a non-interactive proof P that the tally is correct. A *tally* is a vector $\mathbf{X}$ of length n_C such that $\mathbf{X}[j]$ indicates the number of votes for candidate c_j.[8]
- **Verify**, denoted $v \leftarrow \mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k)$, can be executed by anyone to audit the election. Verify takes

3. https://github.com/benadida/helios-server/tree/master/helios_auth/auth_systems, accessed 4 Aug 2015.

4. Let $\mathsf{Alg}(in; r)$ denote the output of probabilistic algorithm Alg on input in and random coins r. Let $\mathsf{Alg}(in)$ denote $\mathsf{Alg}(in; r)$, where r is chosen uniformly at random. And let $\leftarrow$ denote assignment.

5. Some election schemes (e.g., Helios, Helios-C, and JCJ) permit the tallier's role to be distributed amongst several talliers. For simplicity, we consider only a single tallier in this paper.

6. The maximum ballots and candidate numbers are used to formalize Correctness. Helios requires that the maximum number of ballots is less than or equal to the size of the underlying encryption scheme's message space, and JCJ requires that the maximum number of candidates is less than or equal to the size of the underlying encryption scheme's message space.

7. Bulletin boards have also been modeled as public broadcast channels [48], [105], [108]. We abstract from the details of channels by employing sets to represent the data sent on them. We favor sets over multisets, because Cortier and Smyth [44], [45] demonstrate attacks against privacy when the bulletin board is modeled as a multiset.

8. Let $\mathbf{X}[i]$ denote component i of vector $\mathbf{X}$.

as input the public key $PK_{\mathcal{T}}$ of the tallier, the bulletin board BB, the number of candidates n_C, a tally $\mathbf{X}$, a proof P of correct tallying, and security parameter k. It outputs a bit v, which is 1 if the tally successfully verifies and 0 otherwise. We assume that Verify is deterministic.

Election schemes must satisfy Correctness, which asserts that tallies produced by Tally corresponds to the choices input to Vote:

Definition 1 (Correctness)**.** *There exists a negligible function μ, such that for all security parameters k, integers n_B and n_C, and choices $\beta_1, \ldots, \beta_{n_B} \in \{1, \ldots, n_C\}$, it holds that if $\mathbf{Y}$ is a vector of length n_C whose components are all 0, then*

$$\begin{aligned}
&\Pr[(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k);\\
&\quad \textbf{for } 1 \leq i \leq n_B \textbf{ do}\\
&\qquad b_i \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta_i, k);\\
&\qquad \mathbf{Y}[\beta_i] \leftarrow \mathbf{Y}[\beta_i] + 1;\\
&\quad BB \leftarrow \{b_1, \ldots, b_{n_B}\};\\
&\quad (\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, k) :\\
&\quad n_B \leq m_B \wedge n_C \leq m_C \Rightarrow \mathbf{X} = \mathbf{Y}] > 1 - \mu(k).
\end{aligned}$$

Note that Correctness does not involve an adversary. Correctness therefore stipulates that, under ideal conditions, an election scheme does indeed produce the correct tally. Correctness is not actually necessary to achieve verifiability: our definition of universal verifiability will ensure that, in the presence of an adversary, Verify detects any errors in the tally. But it is reasonable to rule out election schemes that simply do not work properly under ideal conditions.

Election schemes must also satisfy Completeness, which stipulates that tallies produced by Tally will actually be accepted by Verify:

Definition 2 (Completeness)**.** *There exists a negligible function μ, such that for all security parameters k, bulletin boards BB, and integers n_C, it holds that*

$$\begin{aligned}
&\Pr[(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k);\\
&\quad (\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, k) :\\
&\quad |BB| \leq m_B \wedge n_C \leq m_C \Rightarrow\\
&\qquad \mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1] > 1 - \mu(k).
\end{aligned}$$

Without Completeness, election schemes might be vulnerable to biasing attacks, as we show in Section VII-B.

Finally, election schemes must satisfy Injectivity, which asserts that a ballot cannot be interpreted as a vote for more than one candidate:

Definition 3 (Injectivity)**.** *For all security parameters k, public keys $PK_{\mathcal{T}}$, integers n_C, and choices β and β', such that $\beta \neq \beta'$, we have*

$$\begin{aligned}
&\Pr[b \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k);\\
&\quad b' \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta', k) :\\
&\quad b \neq \perp \wedge b' \neq \perp \Rightarrow b \neq b'] = 1.
\end{aligned}$$

Injectivity ensures that distinct choices are not mapped by Vote to the same ballot. Without Injectivity, an election scheme might produce ballots whose meaning is ambiguous. For example, if $\mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r)$ were defined to be $\beta + r$, then a ballot b could be tallied as any well-formed choice β' such that $\beta' = b - r'$ for some r'. But that definition of Vote is prohibited by Injectivity. Thus, Injectivity helps to ensure that the choices used to construct ballots can be uniquely tallied.

Limitations: Our model of election schemes is sufficient to analyze Helios and, after we extend the model to handle internal authentication in Section IV-A, Helios-C and JCJ. These are notable schemes, and formally analyzing their verifiability is a novel contribution. But there are other notable schemes that fall outside our model:

- Pret à Voter [33], MarkPledge [100], Scantegrity II [30], and Remotegrity [122] all rely on features implemented with paper, such as scratch-off surfaces and detachable columns.
- Everlasting privacy [98], which requires Vote to output a public ballot and a secret proof, involving temporal information, to the voter.
- Scytl's Pnyx.core ODBP 1.0 [36], which requires the bulletin board to be divided into two parts: a public part visible to all participants, and a secret part visible only to election administrators.

We leave extension of our model to other election schemes as future work.

B. Election verifiability

Election verifiability comprises three aspects: individual, universal, and eligibility verifiability. We express each as an *experiment*, which is an algorithm that outputs 0 or 1. The adversary *wins* an experiment by causing it to output 1.

1) Individual verifiability: In our model of election schemes, all recorded ballots are posted on the bulletin board. So for a voter to verify that their ballot has been recorded, it suffices to enable them to uniquely identify their ballot on the bulletin board.[9]

Individual verifiability experiment $\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k)$, where Π denotes an election scheme, $\mathcal{A}$ denotes the adversary, and k denotes a security parameter, therefore challenges $\mathcal{A}$ to generate a scenario in which the voter cannot uniquely identify their ballot. In essence, Exp-IV-Ext challenges $\mathcal{A}$ to generate a collision from Vote.[10] If $\mathcal{A}$ cannot win, then voters can uniquely identify their ballots on the bulletin board:

$\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k) =$

9. Section IX addresses the complementary issue of whether a recorded ballot corresponds to the candidate choice a voter intended to make.

10. Exp-IV-Ext can be equivalently formulated as an experiment that challenges $\mathcal{A}$ to predict the output of Vote. See Appendix B for details.

1 $(PK_{\mathcal{T}}, n_C, \beta, \beta') \leftarrow \mathcal{A}(k)$;
2 $b \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$;
3 $b' \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta', k)$;
4 **if** $b = b' \wedge b \neq \bot \wedge b' \neq \bot$ **then**
5 | **return** 1
6 **else**
7 | **return** 0

Line 1 asks $\mathcal{A}$ to compute two candidate choices β and β', such that ballots b and b' for those choices, as computed by Vote in lines 2 and 3, are equal. Individual verifiability thus resembles Injectivity, but individual verifiability allows choices to be equal and allows $\mathcal{A}$ to choose election parameters.

One way to achieve individual verifiability is to base the election scheme on a probabilistic encryption scheme, such as El Gamal [58]. Intuitively, if Vote encrypts the choice using random coins, then it is overwhelmingly unlikely that two votes will result in the same ballot. Our proofs that Helios, Helios-C and JCJ satisfy individual verifiability are based on this idea.

Clash attacks: In a *clash attack* [95], the adversary convinces some voters that a single ballot belongs to all of them. Some clash attacks are possible because of vulnerabilities in the design of Vote. For example, if Vote simply outputs candidate choice β, then a voter has no way to distinguish their vote for β from another voter's vote for β. Exp-IV-Ext detects clash attacks resulting from vulnerabilities in Vote.

Some clash attacks, however, are possible because the adversary subverts the implementation of Vote. For example, the adversary might replace some hardware or software, or compromise the random number generator. If any one of these aspects is compromised, then Vote has effectively been changed to a different algorithm Vote$'$. The conclusions drawn by a security analyst who uses our definition of individual verifiability to analyze Vote would not necessarily be applicable to Vote$'$.

In short, a voter can verify that their ballot has been recorded if and only if they run the correct Vote algorithm. We make no guarantees to voters that do not run the correct Vote algorithm. One way to make stronger guarantees is to use cut-and-choose protocols to audit ballots [15], [16]. This would require modeling voting as an interactive protocol with the adversary, rather than as an algorithm. We leave this extension as future work.

2) Universal verifiability: For an election to be universally verifiable, anyone must be able to check that a tally is correct with respect to recorded ballots—that is, the tally represents the choices used to construct the recorded ballots. Because anyone can execute Verify, it suffices that Verify accepts only when that property holds.

Universal verifiability experiment $\mathsf{Exp\text{-}UV\text{-}Ext}(\Pi, \mathcal{A}, k)$ therefore challenges adversary $\mathcal{A}$ to concoct a scenario in which Verify incorrectly accepts:

$\mathsf{Exp\text{-}UV\text{-}Ext}(\Pi, \mathcal{A}, k) =$

1 $(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) \leftarrow \mathcal{A}(k)$;
2 $\mathbf{Y} \leftarrow \mathit{correct\text{-}tally}(PK_{\mathcal{T}}, BB, n_C, k)$;
3 **if** $\mathbf{X} \neq \mathbf{Y} \wedge \mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$ **then**
4 | **return** 1
5 **else**
6 | **return** 0

In line 1, $\mathcal{A}$ is challenged to create a bulletin board BB and purported tally $\mathbf{X}$ of that bulletin board. Line 2 constructs the correct tally $\mathbf{Y}$ of BB, and line 3 checks whether Verify accepts an incorrect tally. If $\mathcal{A}$ cannot win Exp-UV-Ext, then Verify will not accept incorrect tallies. In particular, no ballots can be omitted from the tally, and at most one candidate choice can be included in the tally for each ballot.

Let function *correct-tally* be defined such that for all $PK_{\mathcal{T}}$, BB, n_C, k, ℓ, and $\beta \in \{1, \ldots, n_C\}$,

$$\begin{aligned} &\mathit{correct\text{-}tally}(PK_{\mathcal{T}}, BB, n_C, k)[\beta] = \ell \\ \iff\ &\exists^{=\ell} b \in (BB \setminus \{\bot\}) : \\ &\qquad \exists r : b = \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r). \end{aligned}$$

The vector produced by *correct-tally* must be of length n_C. Component β of vector $\mathit{correct\text{-}tally}(PK_{\mathcal{T}}, BB, n_C, k)$ equals ℓ iff there exist[11] ℓ ballots on the bulletin board that are votes for candidate β. It follows that the output of *correct-tally* represents the choices used to construct the recorded ballots. Note that, without Injectivity, the existential quantification in *correct-tally* could permit a ballot to be tallied for more than one candidate. Of course, *correct-tally* cannot be computed by a PPT algorithm for typical cryptographic election schemes. But that does not matter, because *correct-tally* is never actually computed as part of an election scheme—its use is solely in the definition of Exp-UV-Ext.[12]

Security analysts must convince themselves that *correct-tally* is indeed correct. Because of the function's simplicity, this should be relatively straightforward. By comparison, Tally algorithms for real voting schemes tend to be complicated. For example, compare the complexity of *correct-tally* to Helios's Tally algorithm, which appears in Figure 1 of Appendix C.

By design, Exp-UV-Ext assumes that the ballots on bulletin board BB are exactly the ballots that should be tallied. The external authentication mechanism is assumed to prohibit unauthorized ballots from being posted on BB. Helios makes such an assumption about its external authentication mechanism.

3) Eligibility verifiability: For an election to satisfy eligibility verifiability, anyone must be able to check that every tallied vote was cast by an authorized voter—hence, it must be possible to authenticate ballots. In election schemes with

11. The definition of *correct-tally* employs a *counting quantifier* [110] denoted $\exists^{=}$. Predicate $(\exists^{=\ell} x : P(x))$ holds exactly when there are ℓ distinct values for x such that $P(x)$ is satisfied. Variable x is bound by the quantifier, whereas ℓ is free.

12. Kiayias et al. [87] use a similar super-polynomial *vote extractor* to recover choices from ballots in an experiment defining verifiability.

external authentication, a trusted third party authenticates ballots. That third party might convince itself that all tallied ballots have been authenticated, but it cannot convince all other parties. Eligibility verifiability, therefore, is not achievable in election schemes with external authentication.

4) Election verifiability: With Exp-IV-Ext and Exp-UV-Ext, we define election verifiability with external authentication. Let a PPT adversary's *success* $\mathsf{Succ}(\mathsf{Exp}(\cdot))$ in an experiment $\mathsf{Exp}(\cdot)$ be the probability that the adversary wins—that is, $\mathsf{Succ}(\mathsf{Exp}(\cdot)) = \Pr[\mathsf{Exp}(\cdot) = 1]$.

Definition 4 (Ver-Ext)**.** *An election scheme* Π *satisfies* election verifiability with external authentication (Ver-Ext) *if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, it holds that* $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k)) + \mathsf{Succ}(\mathsf{Exp\text{-}UV\text{-}Ext}(\Pi, \mathcal{A}, k)) \leq \mu(k)$.

An election scheme satisfies individual verifiability if $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k)) \leq \mu(k)$, and similarly for universal verifiability.

C. Example—Toy scheme from nonces

A toy election scheme satisfying Ver-Ext can be based on nonces. Each voter publishes a nonce paired with her choice of candidate to the bulletin board. This scheme illustrates the essence of election verifiability, even though it does not offer any privacy.

Definition 5. *Election scheme* Nonce *is defined as follows:*

- $\mathsf{Setup}(k)$ *outputs* $(\perp, \perp, p_1(k), p_2(k))$*, where* p_1 *and* p_2 *may be any polynomial functions.*
- $\mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$ *selects a nonce* r *uniformly at random from* $\mathbb{Z}_{2^k}$ *and outputs* (r, β).
- $\mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, k)$ *computes a vector* $\mathbf{X}$ *of length* n_C*, such that* $\mathbf{X}$ *is a tally of the votes on* BB *for which the nonce is in* $\mathbb{Z}_{2^k}$*, and outputs* $(\mathbf{X}, \perp)$.
- $\mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k)$ *outputs* 1 *if* $(\mathbf{X}, P) = \mathsf{Tally}(\perp, BB, n_C, k)$*, and* 0 *otherwise.*

Proposition 1. Nonce *satisfies* Ver-Ext.

Proof sketch. Nonce satisfies individual verifiability, because voters can use their nonce to check that their own ballot appears on the bulletin board. With overwhelming probability, Vote will select unique nonces for each voter, hence generate distinct ballots. Nonce also satisfies universal verifiability, because plaintext candidate choices are posted on the bulletin board. □

D. Orthogonality

Exp-IV-Ext and Exp-UV-Ext capture orthogonal security properties. A scheme that satisfies individual verifiability but violates universal verifiability can be constructed from Nonce by modifying Verify to always output 1. Voters can still check that their own ballot appears. But an adversary can easily win Exp-UV-Ext, because Verify will accept any tally. A scheme that satisfies universal verifiability but violates individual verifiability can be constructed from Nonce by removing the nonces, leaving just the voter's choice in the ballots. Call that scheme Choice. Anyone can still verify the tally of the election, but an adversary can easily win Exp-IV-Ext, because two votes for the same candidate will collide.

III. Case Study: Helios

Helios [5] is an open-source, web-based electronic voting system,[13] which has been deployed in the real-world. The International Association of Cryptologic Research (IACR) has used Helios annually since 2010 to elect board members [18], [70], [77], the Catholic University of Louvain used Helios to elect the university president [5], and Princeton University has used Helios to elect several student governments [3], [102].

Helios is intended to satisfy verifiability whilst maintaining *ballot secrecy*—i.e., without revealing voters' votes. For ballot secrecy, voters encrypt candidate choices using a homomorphic encryption scheme, these encrypted choices are homomorphically combined, and the tallier decrypts the homomorphic combination to reveal the tally.[14] For verifiability, encryption and decryption steps are accompanied by zero-knowledge proofs.

Informally, Helios works as follows:

- **Setup.** The tallier generates a key pair for a homomorphic encryption scheme and publishes the public key.
- **Voting.** A voter encrypts her candidate choice with the tallier's public key, and proves in zero-knowledge that the ciphertext contains a well-formed choice. The voter posts her ballot (i.e., ciphertext and proof) on the bulletin board. (The bulletin board is assumed to correctly authenticate voters during posting.)
- **Tallying.** The tallier discards any ballots from the bulletin board for which proofs do not hold. The tallier homomorphically combines the ciphertexts in the remaining ballots, decrypts the homomorphic combination, and proves in zero-knowledge that decryption was performed correctly. Finally, the tallier publishes the winning candidate and proof of correct decryption.
- **Verification.** A verifier recomputes the homomorphic combination and checks all the zero-knowledge proofs.

Helios was first implemented as *Helios 2.0*.[15,16]

Vulnerabilities have been discovered against Helios 2.0, and mitigations against those vulnerabilities have been proposed.

13. https://vote.heliosvoting.org/, accessed 16 Nov 2015.

14. Homomorphic combination of ciphertexts is straightforward for two-candidate elections [14], [19], [38], [74], [107], since choices (e.g., "yes" or "no") can be encoded as 1 or 0. Multi-candidate elections are also possible [19], [52], [73].

15. https://github.com/benadida/helios/releases/tag/2.0, released 25 Jul 2009, accessed 16 Nov 2015.

16. Helios 2.0 builds upon *Helios 1.0* [2]. But, the two systems are rather different. In particular, the Helios 2.0 tallier homomorphically combines encrypted choices and decrypts the homomorphic combination to reveal the tally, whereas the Helios 1.0 tallier mixes encrypted choices and decrypts the ciphertexts output by the mix.

- Verifiability exploits are attributed to application of the Fiat–Shamir transformation without inclusion of statements in hashes (i.e., the weak Fiat–Shamir transformation), and including statements in hashes (i.e., applying the Fiat–Shamir transformation) is postulated as a defense [21].
- Ballot secrecy exploits are attributed to tallying meaningfully related ballots,[17] and omitting such ballots from the tally (i.e., ballot weeding) is postulated as a defense [44], [45], [112], [115], [116], [118].

Given the verifiability exploits, we would not expect $\mathsf{Ver\text{-}Ext}$ to hold for Helios 2.0. Indeed, we formalize a generic construction for Helios-like election schemes (Appendix C), which we instantiate to derive a formal description of Helios 2.0 (Appendix D). And using that description, we can prove that Helios 2.0 is not verifiable:

Proposition 2. *Helios 2.0 does not satisfy* $\mathsf{Ver\text{-}Ext}$.

The proof of Proposition 2 appears in Appendix D.

The specification for the next Helios release (Helios'12) is intended to mitigate against vulnerabilities [4].[18] In particular, it incorporates the Fiat–Shamir transformation (rather than the weak Fiat–Shamir transformation). And it incorporates ballot weeding: any ballot containing a previously observed hash is omitted from the tally. Although ballot weeding can be sufficient for ballot secrecy (cf. [113, §6]), we have found that it violates universal verifiability. In particular, an adversary can observe a voter's ballot and cast a related ballot, such that the voter's ballot is omitted from tallying. (This could be achieved, for example, by manipulating the bulletin board to ensure observation of the adversary's ballot before the voter's ballot, since this causes the voter's ballot to be weeded.) Our definition of universal verifiability requires all ballots on the bulletin board to be tallied, thus it is violated by ballot weeding. It follows that Helios'12 does not satisfy $\mathsf{Ver\text{-}Ext}$, because that scheme relies upon ballot weeding to defend against ballot secrecy violations.

Remark 3. *Helios'12 does not satisfy* $\mathsf{Ver\text{-}Ext}$.

Proof sketch. Helios'12 uses ballot weeding, which violates universal verifiability, as described above. □

An informal proof of Remark 3 follows immediately from our discourse. A formal proof would require a formal description of Helios'12. Such a formal description can be derived as a straight-forward variant of Helios 2.0 that applies the Fiat–Shamir transformation (rather than the weak Fiat–Shamir transformation) and uses ballot weeding. These details provide little value, so we do not pursue them further.

To ensure universal verifiability, we propose Helios'16, a variant of Helios'12. Our variant defends against ballot secrecy violations by incorporating proposals by Smyth et al. [119] and Smyth [113] for non-malleable ballots, rather than proposals for ballot weeding. We give a formal description of Helios'16 in Appendix E. Using that formalization, we can prove that Helios'16 is verifiable:

Theorem 4. *Helios'16 satisfies* $\mathsf{Ver\text{-}Ext}$.

Proof sketch. Helios'16 satisfies individual verifiability, because the probabilistic encryption scheme ensures that ballots are unique, with overwhelming probability. And Helios'16 satisfies universal verifiability, because the zero-knowledge proofs can be publicly verified. □

A formal proof of Theorem 4 appears in Appendix F. The proof assumes the random oracle model [11]. This proof provides strong motivation for future Helios releases being based upon Helios'16.

IV. Internal Authentication

Some election schemes implement their own authentication mechanisms. JCJ [81]–[83] and Civitas [37], for example, authenticate ballots based on *credentials* issued to voters by a registration authority. Schemes with this kind of internal authentication enable verification of whether tallied ballots were cast by authorized voters.

A. Election scheme

A *registrar* is responsible for issuing authentication *credentials* to voters.[19] Each voter is associated with a credential pair (pk, sk). The voter uses private credential sk to construct a ballot. Public credential pk is used during tallying and verification. Let L denote the *electoral roll*, which is the set of all public credentials.

An *election scheme with internal authentication*, which henceforth in this section we abbreviate as "election scheme," is a tuple $(\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ of PPT algorithms. The algorithms are now denoted as follows:

- $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$
- $(pk, sk) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$
- $b \leftarrow \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k)$
- $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, L, n_C, k)$
- $v \leftarrow \mathsf{Verify}(PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k)$

Setup is unchanged from election schemes with external authentication (cf. §II-A). The only change to Vote is that it now accepts private credential sk as input. Similarly, the only change to Tally and Verify is that they now accept electoral roll L as input. $\mathsf{Register}$ is executed by the registrar. It takes as input the public key $PK_{\mathcal{T}}$ of the tallier and security parameter k, and it outputs a *credential pair* (pk, sk). After all voters have been registered, the registrar certifies the electoral roll, perhaps by digitally signing and publishing it.[20]

17. Meaningfully related ballots can be constructed because Helios ballots are malleable.

18. The current version of Helios, *Helios 3.1.4* (https://github.com/benadida/helios-server/releases/tag/v3.1.4, released 10 Mar 2011, accessed 19 Aug 2015), predates the discovery of verifiability exploits, hence, it is vulnerable.

19. Some election schemes (e.g., Helios-C and JCJ) permit the registrar's role to be distributed among several registrars. For simplicity, we consider only a single registrar in this paper.

20. It might seem surprising that $\mathsf{Register}$ does not require the registrar to provide any private keys as input. But in constructions of election schemes with internal authentication, e.g., [37], [83], the registrar does not sign credential pairs with its own private key. Rather, the registrar signs the electoral roll.

Election schemes must continue to satisfy Correctness, Completeness, and Injectivity, which we update to include private credentials and the electoral roll:

Definition 6 (Correctness)**.** *There exists a negligible function μ, such that for all security parameters k, integers n_B and n_C, and choices $\beta_1, \ldots, \beta_{n_B} \in \{1, \ldots, n_C\}$, it holds that if $\mathbf{Y}$ is a vector of length n_C whose components are all 0, then*

$\Pr[(PK_\mathcal{T}, SK_\mathcal{T}, m_B, m_C) \leftarrow \mathsf{Setup}(k);$
for $1 \leq i \leq n_B$ **do**
$\quad (pk_i, sk_i) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k);$
$\quad b_i \leftarrow \mathsf{Vote}(sk_i, PK_\mathcal{T}, n_C, \beta_i, k);$
$\quad \mathbf{Y}[\beta_i] \leftarrow \mathbf{Y}[\beta_i] + 1;$
$L \leftarrow \{pk_1, \ldots, pk_{n_B}\};$
$BB \leftarrow \{b_1, \ldots, b_{n_B}\};$
$(\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_\mathcal{T}, BB, L, n_C, k):$
$n_B \leq m_B \wedge n_C \leq m_C \Rightarrow \mathbf{X} = \mathbf{Y}] > 1 - \mu(k).$

Definition 7 (Completeness)**.** *There exists a negligible function μ, such that for all security parameters k, bulletin boards BB, and integers n_C and n_V, it holds that*

$$\begin{aligned}&\Pr[(PK_\mathcal{T}, SK_\mathcal{T}, m_B, m_C) \leftarrow \mathsf{Setup}(k);\\ &\quad \textbf{for } 1 \leq i \leq n_V \textbf{ do } (pk_i, sk_i) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k);\\ &\quad L \leftarrow \{pk_1, \ldots, pk_{n_V}\};\\ &\quad (\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_\mathcal{T}, BB, L, n_C, k):\\ &\quad |BB| \leq m_B \wedge n_C \leq m_C \Rightarrow\\ &\qquad \mathsf{Verify}(PK_\mathcal{T}, BB, L, n_C, \mathbf{X}, P, k) = 1] > 1 - \mu(k).\end{aligned}$$

Definition 8 (Injectivity)**.** *For all security parameters k, public keys $PK_\mathcal{T}$, integers n_C, and choices β and β', such that $\beta \neq \beta'$, we have*

$$\begin{aligned}&\Pr[(pk, sk) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k);\\ &\quad (pk', sk') \leftarrow \mathsf{Register}(PK_\mathcal{T}, k);\\ &\quad b \leftarrow \mathsf{Vote}(sk, PK_\mathcal{T}, n_C, \beta, k);\\ &\quad b' \leftarrow \mathsf{Vote}(sk', PK_\mathcal{T}, n_C, \beta', k):\\ &\quad b \neq \perp \wedge b' \neq \perp \Rightarrow b \neq b'] = 1.\end{aligned}$$

B. Election verifiability

Recall (from §II-B) that election verifiability is expressed with experiments, and that an adversary wins by causing an experiment to output 1. We henceforth assume that the adversary is *stateful*—that is, information persists across invocations of the adversary in a single experiment. Our experiments in Section II did not need this assumption, because they never invoked the adversary more than once.

In our experiments, below, we model an adversary who cannot corrupt the registration process that issues credentials to voters.[21] Hence our definitions will not detect attacks against verifiability that result solely from weaknesses in the registration process. Secure construction of electoral rolls is not a topic that electronic voting usually addresses—though it seems an important part of any real-world deployment.

1) Individual verifiability: The individual verifiability experiment again challenges adversary $\mathcal{A}$ to generate a scenario in which the voter could not uniquely identify their ballot:[22]

$\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}, k) =$
1 $(PK_\mathcal{T}, n_V) \leftarrow \mathcal{A}(k);$
2 **for** $1 \leq i \leq n_V$ **do** $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k)$
3 $L \leftarrow \{pk_1, \ldots, pk_{n_V}\};$
4 $Crpt \leftarrow \emptyset;$
5 $(n_C, \beta, \beta', i, j) \leftarrow \mathcal{A}^C(L);$
6 $b \leftarrow \mathsf{Vote}(sk_i, PK_\mathcal{T}, n_C, \beta, k);$
7 $b' \leftarrow \mathsf{Vote}(sk_j, PK_\mathcal{T}, n_C, \beta', k);$
8 **if** $b = b' \wedge b \neq \perp \wedge b' \neq \perp \wedge i \neq j \wedge sk_i \notin Crpt \wedge sk_j \notin Crpt$ **then**
9 $\quad$ **return** 1
10 **else**
11 $\quad$ **return** 0

The main differences from the corresponding experiment for external authentication (§II-B1) are that voters are registered in line 2, and that $\mathcal{A}$ is given access to an oracle C in line 5. The oracle is used to model $\mathcal{A}$ corrupting voters and learning their private credentials: on invocation $C(\ell)$, where $1 \leq \ell \leq n_V$, the oracle records that voter ℓ is corrupted by updating $Crpt$ to be $Crpt \cup \{sk_\ell\}$ and outputs sk_ℓ. In line 5, the voter indices output by $\mathcal{A}$ must be legal with respect to n_V, but we elide that detail from the experiment for simplicity. Line 8 ensures that $\mathcal{A}$ cannot trivially win by corrupting voters.

2) Universal verifiability: The universal verifiability experiment again challenges $\mathcal{A}$ to concoct a scenario in which Verify incorrectly accepts:

$\mathsf{Exp\text{-}UV\text{-}Int}(\Pi, \mathcal{A}, k) =$
1 $(PK_\mathcal{T}, n_V) \leftarrow \mathcal{A}(k);$
2 **for** $1 \leq i \leq n_V$ **do** $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k)$
3 $L \leftarrow \{pk_1, \ldots, pk_{n_V}\};$
4 $M \leftarrow \{(pk_1, sk_1), \ldots, (pk_{n_V}, sk_{n_V})\};$
5 $(BB, n_C, \mathbf{X}, P) \leftarrow \mathcal{A}(M);$
6 $\mathbf{Y} \leftarrow \textit{correct-tally}(PK_\mathcal{T}, BB, M, n_C, k);$
7 **if** $\mathbf{X} \neq \mathbf{Y} \wedge \mathsf{Verify}(PK_\mathcal{T}, BB, L, n_C, \mathbf{X}, P, k) = 1$ **then**
8 $\quad$ **return** 1
9 **else**
10 $\quad$ **return** 0

The main differences from the corresponding experiment for external authentication (§II-B2) are that voters are registered in line 2, and their credential pairs are used in the rest of the experiment.

The tally of recorded ballots should contain at most one vote per voter. Hence, election schemes must handle *revotes*—i.e., multiple ballots submitted by the same voter. Election schemes

21. Küsters and Truderung [91] explore some consequences of permitting adversarial influence during registration.

22. Unlike Exp-IV-Ext, a variant of Exp-IV-Int that challenges $\mathcal{A}$ to predict the output of Vote is strictly stronger. See Appendix B for details.

with external authentication implicitly handle revoting, by assuming a third party ensures that the recorded ballots contain at most one ballot per voter. Election schemes with internal authentication must explicitly handle revoting by tallying only authorized ballots. A ballot is *authorized* if it is constructed with a private credential from M, and that private credential was not used to construct any other ballot on BB.[23,24]

Function *correct-tally* is now modified to tally only authorized ballots: let function *correct-tally* now be defined such that for all $PK_{\mathcal{T}}$, BB, M, n_C, k, ℓ, and $\beta \in \{1, \ldots, n_C\}$,

$$\begin{aligned}&\mathit{correct\text{-}tally}(PK_{\mathcal{T}}, BB, M, n_C, k)[\beta] = \ell \\ &\iff \exists^{=\ell} b \in \mathit{authorized}(PK_{\mathcal{T}}, (BB \setminus \{\perp\}), M, n_C, k) : \\ &\qquad \exists sk, r : b = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k; r).\end{aligned}$$

By comparison, the original *correct-tally* function (§II-B2) tallies all the ballots on BB.

Let *authorized* be defined as follows:

$$\begin{aligned}&\mathit{authorized}(PK_{\mathcal{T}}, BB, M, n_C, k) = \\ &\{b : b \in BB \\ &\quad \wedge \exists pk, sk, \beta, r : b = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k; r) \\ &\qquad \wedge (pk, sk) \in M \wedge \neg \exists b', \beta', r' : b' \in (BB \setminus \{b\}) \\ &\qquad\quad \wedge b' = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta', k; r')\}.\end{aligned}$$

Function *authorized* discards ballots submitted under the same credential—that is, if there is more than one ballot submitted with a private credential sk, then all ballots submitted under that credential are discarded. Therefore, election schemes that permit revoting cannot by analyzed with this definition of *authorized*. But alternative definitions of *authorized* are possible—for example, if ballots were timestamped, *authorized* could discard all but the most recent ballot submitted under a particular credential.

3) Eligibility verifiability: Recall (from §II-B3) that for an election scheme to satisfy eligibility verifiability, anyone must be able to check that every tallied vote was cast by an authorized voter—hence, it must be possible to authenticate ballots. Because voters are issued credential pairs that can be used to authenticate ballots, it suffices to ensure that knowledge of a private credential is necessary to construct an authentic ballot.

Eligibility verifiability experiment Exp-EV-Int therefore challenges $\mathcal{A}$ to produce a ballot under a private credential that $\mathcal{A}$ does not know:

$\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, k) =$

1 $(PK_{\mathcal{T}}, n_V) \leftarrow \mathcal{A}(k)$;
2 **for** $1 \leq i \leq n_V$ **do** $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$;
3 $L \leftarrow \{pk_1, \ldots, pk_{n_V}\}$;
4 $\mathit{Crpt} \leftarrow \emptyset$; $\mathit{Rvld} \leftarrow \emptyset$;
5 $(n_C, \beta, i, b) \leftarrow \mathcal{A}^{C,R}(L)$;
6 **if** $\exists r : b = \mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k; r) \wedge b \neq \perp \wedge b \notin \mathit{Rvld} \wedge sk_i \notin \mathit{Crpt}$ **then**
7 **return** 1
8 **else**
9 **return** 0

In line 1, $\mathcal{A}$ chooses the tallier's public key and the number of voters. Line 2 registers voters. $\mathcal{A}$ is not permitted to influence registration while it is in progress. In particular, $\mathcal{A}$ is not permitted to choose credential pairs, because by doing so $\mathcal{A}$ could trivially win the experiment.

Line 4 initializes two sets: *Crpt* is a set of voters who have been corrupted, meaning that $\mathcal{A}$ has learned their private credential, and *Rvld* is a set of ballots that have been revealed to $\mathcal{A}$. The former set models $\mathcal{A}$ coercing voters to reveal their private credentials. The latter set models $\mathcal{A}$ observing ballots on the bulletin board.

Line 5 challenges $\mathcal{A}$ to produce a ballot b with the help of two oracles. Oracle C is the same oracle as in Exp-IV-Int (cf. §IV-B1); it leaks the private credentials of corrupted voters to $\mathcal{A}$. Oracle R reveals ballots. On invocation $R(i, \beta, n_C)$, where $1 \leq i \leq n_V$, oracle R does the following:

- Computes a ballot b that represents a vote for candidate β by a voter with private credential sk_i, that is, computes $b \leftarrow \mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k)$.
- Records b as being revealed by updating Rvld to be $\mathit{Rvld} \cup \{b\}$.
- Outputs b.

In line 6, $\mathcal{A}$ wins if (i) the ballot is *authentic*, meaning that it is the output of Vote on an authorized credential, and (ii) that credential belongs to a voter that $\mathcal{A}$ did not corrupt, and (iii) that ballot was not revealed. If $\mathcal{A}$ cannot succeed in this experiment, then only authorized votes are tallied.

4) Election verifiability: With Exp-IV-Int, Exp-UV-Int, and Exp-EV-Int, we define election verifiability with internal authentication.

Definition 9 (Ver-Int)**.** *An election scheme* Π *satisfies* election verifiability with internal authentication (Ver-Int) *if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, it holds that* $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}, k)) + \mathsf{Succ}(\mathsf{Exp\text{-}UV\text{-}Int}(\Pi, \mathcal{A}, k)) + \mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, k)) \leq \mu(k)$.

An election scheme satisfies eligibility verifiability if $\mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, k)) \leq \mu(k)$, and similarly for individual and universal verifiability.

23. Helios-C is claimed to support an alternative definition of authorized, whereby only the last ballot cast by a voter is authorized. We found that Helios-C does not support this definition. In particular, an adversary can observe the ballots cast by a voter and replay one of those ballots. The replayed ballot will overwrite the last ballot cast by the voter and will be authorized instead of it.

24. JCJ is claimed to support alternative definitions of authorized—e.g., only the last ballot cast by a voter is authorized—using a policy [83, §4.1]. We found that the policy proposed by Juels et al. (namely, "order of postings to [the bulletin board]") does not support this definition of authorized. In particular, an adversary can intercept a voter's ballot and replay that ballot after observing the voter's revote, thus the policy incorrectly defines the first ballot as authorized. This could be prevented by proving knowledge of previously constructed ballots (cf. Clarkson et al. [37]).

C. Example—Toy schemes from digital signatures

A toy election scheme satisfying Ver-Int can be based on a digital signature scheme.[25] Each voter publishes their signed candidate choice on the bulletin board.

Definition 10. *Suppose* $\Gamma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *is a digital signature scheme. Let election scheme* $\mathsf{Sig}(\Gamma)$ *be defined as follows:*

- $\mathsf{Setup}(k)$ *outputs* $(\perp, \perp, p_1(k), p_2(k))$*, where* p_1 *and* p_2 *may be any polynomial functions.*
- $\mathsf{Register}(PK_{\mathcal{T}}, k)$ *outputs a key pair produced by* $\mathsf{Gen}(k)$*.*
- $\mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k)$ *outputs a signature produced by* $\mathsf{Sign}(sk, \beta)$*.*
- $\mathsf{Tally}(SK_{\mathcal{T}}, BB, L, n_C, k)$ *computes a vector* $\mathbf{X}$ *of length* n_C*, such that* $\mathbf{X}$ *is a tally of all the ballots on* BB *that are signed by distinct private keys whose corresponding public keys appear in* L *(formally, signatures can be checked using algorithm* Ver*), and outputs* $(\mathbf{X}, \perp)$*.*
- $\mathsf{Verify}(PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k)$ *outputs* 1 *if* $(\mathbf{X}, P) = \mathsf{Tally}(\perp, \perp, BB, L, n_C, \perp)$ *and* 0 *otherwise.*

Let Sig denote $\mathsf{Sig}(\Gamma)$ for an unspecified digital signature scheme Γ satisfying strong unforgeablility [7], [24].[26] The verifiability of Sig follows from the security of the underlying signature scheme:

Proposition 5. Sig *satisfies* Ver-Int.

Proof sketch. Sig satisfies individual verifiability, because voters can verify that their signed choices appear on the bulletin board. Sig satisfies universal verifiability, because signed plaintext choices are posted on BB. Finally, Sig satisfies eligibility verifiability, because anyone can check that the signed choices belong to registered voters. □

D. Orthogonality

Exp-IV-Int, Exp-UV-Int, and Exp-EV-Int capture mostly orthogonal security properties, as shown in Table I. Individual and universal verifiability are orthogonal, and eligibility verifiability implies individual verifiability.

Theorem 6. *If an election scheme* Π *satisfies* Exp-EV-Int*, then* Π *also satisfies* Exp-IV-Int.

Proof sketch. If Π satisfies Exp-EV-Int, then no one can construct a ballot that appears to be associated with public credential pk unless they know private credential sk. That means that a voter can uniquely identify their ballot, because no one else knows their private credential. Therefore Π satisfies Exp-IV-Int. □

The proof of Theorem 6 appears in Appendix G.

In Table I, $\mathsf{AlwaysVerify}(\cdot)$ is a function that transforms an election scheme by compromising Verify to always return 1. Thus, $\mathsf{AlwaysVerify}(\Pi)$ is guaranteed not to satisfy Exp-UV-Int. Similarly, $\mathsf{IgnoreCreds}(\cdot)$ is a function that accepts as input an election scheme with external authentication and returns as output an election scheme with internal authentication. The resulting scheme, however, simply ignores credentials altogether: Register returns $(\perp, \perp)$, Vote ignores sk, and Tally and Verify ignore L. Thus, $\mathsf{IgnoreCreds}(\Pi)$ is guaranteed not to satisfy Exp-EV-Int. Using those functions, we briefly explain each line of the table:

Line	IV	UV	EV	Scheme
1	✗	✗	✗	AlwaysVerify(IgnoreCreds(Choice))
2	✗	✗	✓	—
3	✗	✓	✗	IgnoreCreds(Choice)
4	✗	✓	✓	—
5	✓	✗	✗	AlwaysVerify(IgnoreCreds(Nonce))
6	✓	✗	✓	AlwaysVerify(Sig)
7	✓	✓	✗	Malleable Sig
8	✓	✓	✓	Sig

TABLE I
ELECTION SCHEMES THAT SATISFY EACH COMBINATION OF INDIVIDUAL, UNIVERSAL AND ELIGIBILITY VERIFIABILITY

1) Recall (from §II-D) that Choice is the election scheme in which ballots contain only the plaintext candidate choice. By compromising Verify and ignoring credentials, we obtain a scheme that satisfies no properties.
2) By Theorem 6, this situation is impossible.
3) Compared to line 1 of Table I, this scheme satisfies Exp-UV-Int, because Verify is not compromised.
4) By Theorem 6, this situation is impossible.
5) Nonce satisfies Exp-IV-Ext and Exp-UV-Ext. Moreover, IgnoreCreds(Nonce) satisfies Exp-IV-Int and Exp-UV-Int. By compromising Verify, we obtain a scheme that satisfies only Exp-IV-Int.
6) Sig satisfies all three properties. By compromising Verify, we obtain a scheme that satisfies only Exp-IV-Int and Exp-EV-Int.
7) By making Sig's underlying signature scheme malleable,[27] we could obtain a scheme that does not satisfy Exp-EV-Int, because the adversary could construct a valid ballot out of a revealed ballot. But the scheme would continue to satisfy Exp-IV-Int and Exp-UV-Int.
8) Sig satisfies all three properties.

V. CASE STUDY: HELIOS-C

Helios-C [41], [42] is a variant of Helios (cf. §III) for two-candidate elections in which ballots are digitally signed.[28]

25. Digital signature schemes are defined in Appendix A.

26. Strong unforgeability is defined in Appendix A.

27. Given a message m and signature σ, a *malleable* signature scheme permits computation of a signature σ' on a related message m' [27]. The malleable signature scheme Sig used in line 7 of Table I would need to enable an adversary to transform a signature on a well-formed candidate β into a signature on a distinct, well-formed candidate β'.

Informally, Helios-C works as follows [41, §5]:

- **Setup.** As in Section III.
- **Registration.** To register a voter, the registrar generates a key pair for a signature scheme and sends the private key to the voter. After all voters are registered, the registrar publishes electoral roll L.
- **Voting.** A voter generates a ciphertext and proof as in Section III, signs the ciphertext and proof with their private key, and posts their public key, ciphertext, proof, and signature on the bulletin board.
- **Tallying.** The tallier aborts if any ballots on the bulletin board are not signed by distinct private keys whose corresponding public keys appear in L. The tallier also aborts if there exists a proof on the bulletin board that does not hold. The ciphertexts and proofs are processed as in Section III.
- **Verification.** If the tallier aborted, then a verifier immediately accepts. Otherwise, the tallier recomputes the homomorphic combination and checks all the zero-knowledge proofs, as in Section III.

Whilst analyzing Helios-C, we discovered that aborting violates our definition of universal verifiability. In particular, an adversary could post an ill-formed ballot on the bulletin board. (For example, a malicious tallier could secretly tally the recorded ballots while the election is in progress and, if that tally is unfavorable to the tallier's preferred candidate, then the tallier could post an ill-formed ballot on the bulletin board.) That ballot will cause tallying to abort. And verifiers will accept that abort. Yet, our definition of universal verifiability demands that verifiers only accept outcomes representing all the choices used to construct the recorded ballots, which aborting violates. Thus, Helios-C does not satisfy our definition of universal verifiability.[29] Nonetheless, a variant of Helios-C that disregards ill-formed ballots would satisfy our definition of universal verifiability.

Remark 7. *Helioc-C does not satisfy* $\mathsf{Ver\text{-}Int}$.

Proof sketch. Helios-C aborts on errors in a manner that violates universal verifiability, as described above. □

An informal proof of Remark 7 follows immediately from our discourse and we do not pursue a formal proof.

Cortier et al. [41] analyzed Helios-C using a different definition of universal verifiability. That definition can be satisfied by schemes in which tallying aborts in a manner that anyone will accept. In particular, the experiment used by that definition cannot be won by an adversary that causes an abort. Thus, verifiers accept outcomes that do not include the choices used to construct voters' ballots. By comparison, our definition demands that verifiers reject such outcomes.

VI. Case Study: JCJ

JCJ (named for its designers, Juels, Catalano, and Jakobsson) [81]–[83] is a *coercion-resistant* election scheme, meaning voters cannot prove whether or how they voted, even if they can interact with the adversary while voting. Coercion resistance protects elections from improper influence by adversaries.

To achieve verifiability and coercion resistance, JCJ uses verifiable *mixnets*, which anonymize a set of messages.[30] During tallying, all encrypted choices are anonymized by a mixnet, then all choices are decrypted. The tally is computed from the decrypted choices. Informally, JCJ works as follows:

- **Setup.** The tallier generates a key pair $(PK_{\mathcal{T}}, SK_{\mathcal{T}})$ for an encryption scheme and publishes the public key.
- **Registration.** To register a voter, the registrar generates a nonce, which is sent to the voter and serves as the private credential. The public credential is computed as an encryption of the private credential with $PK_{\mathcal{T}}$. After all voters are registered, the registrar publishes the electoral roll.
- **Voting.** A voter encrypts her candidate choice with $PK_{\mathcal{T}}$. She also encrypts her private credential with $PK_{\mathcal{T}}$. She proves in zero-knowledge that she simultaneously knows both plaintexts, and that her choice is well-formed. The voter posts her ballot (i.e., both ciphertexts and the proof) on the bulletin board.
- **Tallying.** The tallier discards any ballots from the bulletin board for which the zero-knowledge proofs do not verify. All unauthorized ballots are then discarded through a combination of protocols that includes verifiable mixnets and *plaintext equivalence tests* (PETs) [78]. (PETs enable proof that two ciphertexts contain the same plaintext without revealing that plaintext.) The tallier decrypts and publishes the remaining ballots, along with a proof that decryption was performed correctly.
- **Verification.** A verifier checks all the proofs included in ballots, and all the proofs published during tallying.

Appendix H gives a formal description of JCJ. That formalization satisfies individual and universal verifiability, assuming that the cryptographic primitives satisfy certain properties that we identify. But the formalization fails to satisfy eligibility verifiability, because knowledge of the tallier's private key $SK_{\mathcal{T}}$ suffices to construct ballots that appear authentic: with $SK_{\mathcal{T}}$, any public credential can be decrypted to discover the corresponding private credential. Note that $\mathsf{Exp\text{-}EV\text{-}Int}$ permits an adversary $\mathcal{A}$ to choose the tallier's key pair, so $\mathcal{A}$ does know $SK_{\mathcal{T}}$ hence can construct a ballot that suffices to win $\mathsf{Exp\text{-}EV\text{-}Int}$.

We can nonetheless prove that JCJ satisfies a variant of eligibility verifiability. Consider the following experiment,

28. Helios-C has been implemented (https://github.com/glondu/helios-server/tree/heliosc, released c. 2013, accessed 25 Nov 2015), but development has ceased in favour of the *Belenios* variant (https://github.com/glondu/belenios/releases/tag/1.0, released 22 Apr 2016, accessed 25 Apr 2016). We analyse Helios-C because a cryptographic definition has been presented in the literature, whereas Belenios has not appeared in the literature. (Results for one system do not imply results for the other, because the two systems are rather different. And similarly for a further variant [40] of Helios-C.)

29. Helios 2.0, Helios'12 and Helios'16 do not abort, so they are not similarly effected.

30. Chaum [28] introduced mixnets. Adida [1] surveys verifiable mixnets.

which does not permit the adversary to choose the tallier's key pair:

Exp-EV-Int-Weak$(\Pi, \mathcal{A}, k) =$

1 $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$;
2 $n_V \leftarrow \mathcal{A}(PK_{\mathcal{T}}, k)$;
3 **for** $1 \leq i \leq n_V$ **do** $(pk_i, sk_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$;
4 $L \leftarrow \{pk_1, \ldots, pk_{n_V}\}$;
5 $\mathit{Crpt} \leftarrow \emptyset;\ \mathit{Rvld} \leftarrow \emptyset$;
6 $(n_C, \beta, i, b) \leftarrow \mathcal{A}^{C,R}(L)$;
7 **if** $\exists r : b = \mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k; r) \wedge b \neq \perp \wedge b \notin \mathit{Rvld} \wedge sk_i \notin \mathit{Crpt}$ **then**
8 | **return** 1
9 **else**
10 | **return** 0

Line 1 of Exp-EV-Int has been refactored into lines 1 and 2 of Exp-EV-Int-Weak. In line 1 of Exp-EV-Int-Weak, keys are generated by the experiment. In line 2, $\mathcal{A}$ is given the public key but not the private key.

Using Exp-EV-Int-Weak, we define a weaker variant of Ver-Int and prove that JCJ satisfies it:

Definition 11 (Ver-Int-Weak)**.** *An election scheme* Π *satisfies* weak election verifiability with internal authentication (Ver-Int-Weak) *if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, we have* $\mathsf{Succ}(\text{Exp-IV-Int}(\Pi, \mathcal{A}, k)) + \mathsf{Succ}(\text{Exp-UV-Int}(\Pi, \mathcal{A}, k)) + \mathsf{Succ}(\text{Exp-EV-Int-Weak}(\Pi, \mathcal{A}, k)) \leq \mu(k)$.

Theorem 8. *JCJ satisfies* Ver-Int-Weak.

Proof sketch. JCJ satisfies individual verifiability, because the probabilistic encryption scheme ensures that ballots are unique, with overwhelming probability. JCJ satisfies universal verifiability, because the proofs produced throughout tallying can be publicly verified. And JCJ satisfies eligibility verifiability, because $\mathcal{A}$ cannot construct new ballots without knowing a voter's private credential or the tallier's private key. □

A formal proof of Theorem 8 appears in Appendix I. The proof assumes the random oracle model.

The Civitas [37] scheme refines the JCJ scheme. Some refinements relevant to election verifiability are an implementation of a distributed registration protocol, and a mixnet based on randomized partial checking (RPC) [79]. We leave a proof that Civitas satisfies Ver-Int-Weak as future work. In that proof, it would be necessary to assume the RPC construction satisfies the definition of mixnets given in Appendix A. Work by Khazaei and Wikström [85] suggests that actually proving satisfaction is unlikely to be possible. Alternatively, the mixnet could be replaced by one based on zero-knowledge proofs [61], [99].

VII. New classes of attack

Our definitions of election verifiability improve upon existing definitions by detecting two previously unidentified classes of attack:

- *Collusion attacks.* An election scheme's tallying and verification algorithms might be designed such that they collude to accept incorrect tallies.
- *Biasing attacks.* An election scheme's verification algorithm might be designed such that it rejects some legitimate tallies.

Although a well-designed election scheme would hopefully not exhibit these vulnerabilities, it is the job of verifiability definitions to detect malicious schemes, regardless of whether vulnerabilities are due to malice or errors. So definitions of election verifiability should preclude collusion and biasing attacks.

A. Collusion Attacks

Here are two examples of potential collusion attacks:

- **Vote stuffing.** Tally behaves normally, but adds κ votes for candidate β. Verify subtracts κ votes from β, then proceeds with verification as normal. Elections thus verify as normal, except that candidate β receives extra votes.
- **Backdoor tally replacement.** Tally and Verify behave normally, unless a *backdoor* value is posted on the bulletin board BB. For example, if $(SK_{\mathcal{T}}, \mathbf{X}^*)$ appears on BB, then Tally and Verify both ignore the correct tally and instead replace it with tally $\mathbf{X}^*$. Value $SK_{\mathcal{T}}$ is the backdoor here; it cannot appear on BB (except with negligible probability) unless the tallier is malicious.

Vote stuffing is detected by our definitions of Correctness (§II-A and §IV-A), because these definitions require that the tally produced by Tally corresponds to the choices encapsulated in ballots on the bulletin board. Note that vote stuffing is not a failure of eligibility verifiability, because the stuffed votes do not correspond to any ballots on the bulletin board. Backdoor tally replacement is detected by our definitions of universal verifiability (§II-B2 and §IV-B2), because those definitions require Verify to accept only those tallies that correspond to a correct tally of the bulletin board.

We show, next, that the definition of election verifiability by Juels et al. [83] fails to detect vote stuffing and backdoor tally replacement, and that the definition by Cortier et al. [41] fails to detect backdoor tally replacement.

Juels et al. [83] formalize definitions that we name *JCJ-correctness* and *JCJ-verifiability*. JCJ-correctness is intuitively meant to capture that "$\mathcal{A}$ cannot pre-empt, alter, or cancel the votes of honest voters [and] that $\mathcal{A}$ cannot cause voters to cast ballots resulting in double voting" [83, p. 45]; it is formalized in terms of whether the adversary can post ballots on the bulletin board that cause the tally to be computed incorrectly. JCJ-verifiability is intuitively "the ability for any player to check whether the tally... has been correctly computed" [83, p. 46]; it is formalized in terms of whether Verify will accept a tally that differs from the output of Tally. We restate the formal definitions in Appendix J.

To show that the JCJ definitions fail to detect collusion attacks, we first formalize the vote stuffing attack. An election scheme $\Pi = (\ldots, \mathsf{Tally}, \mathsf{Verify})$ can be modified to derive a vote-stuffing election scheme $\mathsf{Stuff}(\Pi, \beta, \kappa) = (\ldots, \mathsf{Tally}_S, \mathsf{Verify}_S)$, which adds κ votes to β, as follows:[31]

- $\mathsf{Tally}_S(\ldots)$ computes $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(\ldots)$ and outputs $(\mathtt{Add}(\mathbf{X}, \beta, \kappa), P)$.[32]
- $\mathsf{Verify}_S(\ldots, \mathbf{X}, P, \ldots)$ computes $v \leftarrow \mathsf{Verify}(\ldots, \mathtt{Add}(\mathbf{X}, \beta, -\kappa), P, \ldots)$ and outputs v.

Proposition 9. *Given an election scheme* Π *satisfying JCJ-correctness and JCJ-verifiability, and integers* β *and* κ*, it holds that* $\mathsf{Stuff}(\Pi, \beta, \kappa)$ *satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in Appendix K.

Second, we formalize the backdoor tally replacement attack. An election scheme $\Pi = (\ldots, \mathsf{Tally}, \mathsf{Verify})$ can be modified to derive an election scheme $\mathsf{Backdoor}(\Pi) = (\ldots, \mathsf{Tally}_B, \mathsf{Verify}_B)$, as follows:

- $\mathsf{Tally}_B(\ldots, SK_{\mathcal{T}}, BB, \ldots)$ checks whether there exists a unique $\mathbf{X}^*$ such that $(SK_{\mathcal{T}}, \mathbf{X}^*) \in BB$. If so, Tally_B outputs $(\mathbf{X}^*, \perp)$. Otherwise, Tally_B computes $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(\ldots, SK_{\mathcal{T}}, BB, \ldots)$ and outputs $(\mathbf{X}, P)$.
- $\mathsf{Verify}_B(PK_{\mathcal{T}}, \ldots, BB, \ldots, \mathbf{X}, P, \ldots)$ checks whether there exists a unique $\mathbf{X}^*$ such that $(SK_{\mathcal{T}}, \mathbf{X}^*) \in BB$.[33] If so, Verify_B outputs 1. Otherwise, Verify_B computes $v \leftarrow \mathsf{Verify}(PK_{\mathcal{T}}, \ldots, BB, \ldots, \mathbf{X}, P, \ldots)$ and outputs v.

Proposition 10. *Given an election scheme* Π *satisfying JCJ-correctness and JCJ-verifiability that does not leak the tallier's private key, it holds that* $\mathsf{Backdoor}(\Pi)$ *satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in Appendix K, where we also formally define key leakage.

Cortier et al. [41] propose definitions similar to *JCJ-verifiability* and insist that election schemes must satisfy their notions of correctness and partial tallying. Vote stuffing is detected by their correctness property, but backdoor tally replacement is not. The ideas remain the same, so we omit formalized results. We have reported these findings to the original authors.[34,35]

B. Biasing attacks

Here are three formalizations of biasing attacks, derived from an election scheme $\Pi = (\ldots, \mathsf{Verify})$.

- **Reject All.** Let $\mathsf{Reject}(\Pi)$ be $(\ldots, \mathsf{Verify}_R)$, where Verify_R always outputs 0. Verify_R therefore always rejects, hence no election can ever be considered valid.
- **Selective Reject.** Let ε be a distinguished value that would not be posted on the bulletin board by honest voters. Let $\mathsf{Selective}(\Pi, \varepsilon)$ be $(\ldots, \mathsf{Verify}_R)$, where $\mathsf{Verify}_R(\ldots, BB, \ldots)$ computes $v \leftarrow \mathsf{Verify}(\ldots, BB, \ldots)$ and outputs 1 if both $v = 1$ and $\varepsilon \notin BB$. Otherwise, Verify_R outputs 0. Verify_R therefore rejects if ε appears on the bulletin board, hence some elections can be invalidated.
- **Biased Reject.** Suppose Z is a set of tallies. Let $\mathsf{Bias}(\Pi, Z)$ be $(\ldots, \mathsf{Verify}_R)$, where $\mathsf{Verify}_R(\ldots, \mathbf{X}, \ldots)$ computes $v \leftarrow \mathsf{Verify}(\ldots, \mathbf{X}, \ldots)$ and outputs 1 if both $v = 1$ and $\mathbf{X} \in Z$. Otherwise, Verify_R outputs 0. Verify_R therefore only accepts a subset of the tallies accepted by Verify, hence biases tallies toward Z.

These formalizations do not satisfy our definition of Completeness (§II-A and §IV-A), hence, our definitions of verifiability detect these biasing attacks.

The definition of verifiability by Juels et al. [83] fails to detect all three of the above attacks, because that definition has no notion of Completeness. For example, it is vulnerable to Biased Reject attacks:

Proposition 11. *Given an election scheme* Π *satisfying JCJ-correctness and JCJ-verifiability, and given a multiset* Z*, it holds that* $\mathsf{Bias}(\Pi, Z)$ *satisfies JCJ-correctness and JCJ-verifiability.*

A formal proof appears in Appendix K.

The definition of verifiability by Kiayias et al. [87] fails to detect Selective Reject attacks, because (like JCJ) the definition has no notion of Completeness. Their notion of Correctness does rule out Reject All and Biased Reject attacks.

Similarly, the definition of verifiability by Cortier et al. [41] detects Biased Reject and Reject All attacks, but fails to detect Selective Reject attacks, because that definition's notion of Completeness does not quantify over all bulletin boards.

VIII. Related Work

Kiayias [86] presents an overview of security properties for election schemes. Many election schemes in the literature state properties called correctness, accuracy, or (universal) verifiability without formally defining those terms.

In the computational model, Juels et al. [81]–[83] and Cortier et al. [41] give game-based definitions of verifiability. Those definitions fail to detect biasing and collusion attacks (cf. §VII). Definitions of universal verifiability (which is just one aspect of election verifiability) in the computational model seem to originate with Benaloh and Tuinstra [17], who define a *correctness* property that says every participant is convinced that the tally is accurate with respect to the votes cast, and with Cohen and Fischer [38], who define *verifiability* to mean that there exists a *check* function that returns good iff the announced tally of the election corresponds to the cast votes.

Kiayias et al. [87] define a property they name *E2E verifiability* (E2E abbreviates "end-to-end"). This property combines our intuitive notions of individual and universal verifiability

31. We omit many of the parameters of Tally and Verify here for simplicity; see Appendix K for details.

32. Let $\mathtt{Add}(\mathbf{X}, \beta, \kappa) = (\mathbf{X}[1], \ldots, \mathbf{X}[\beta - 1], \mathbf{X}[\beta] + \kappa, \mathbf{X}[\beta + 1], \ldots, \mathbf{X}[|\mathbf{X}|])$. And let $|\mathbf{X}|$ denote the length of vector $\mathbf{X}$.

33. Verify_B also needs to check that $SK_{\mathcal{T}}$ is the private key corresponding to $PK_{\mathcal{T}}$. We omit formalizing this detail, but note that it is straightforward for real-world encryption schemes such as El Gamal and RSA.

34. Véronique Cortier and David Galindo, personal communication, Nancy, France, 13 June 2013.

35. David Galindo and Véronique Cortier, email communication, 19 June 2013 & Summer/Autumn 2014.

into a single definition. Their definition fails to detect Selective Reject attacks (cf. §VII). Their definitions, like ours, do not address voter intent—that is, verification by humans that ballots correctly encode candidate choices—as we discuss in Section IX.

Also in the computational model, Groth [68], and Moran and Naor [98], state definitions of verifiability in terms of *universal composability* [26]. These definitions involve defining an *ideal functionality*; part of that is similar to our $\mathit{correct\text{-}tally}$ function. Groth's definition does not guarantee universal verifiability [68, p. 2], but Moran and Naor's does [98, p. 386].

In the symbolic model, Smyth et al. [120] define the first definition of election verifiability. This definition is amenable to automated reasoning, but is stronger than necessary and cannot be satisfied by many election schemes, including Helios and Civitas. Kremer et al. [89] overcome this limitation with a weaker definition that sacrifices amenability to automated reasoning, and Smyth [111, §3] extends this definition. Additionally, the scope of automated reasoning, using the definition by Smyth et al., is limited by analysis tools (e.g., ProVerif [23]), because the function symbols and equational theory used to model cryptographic primitives might not be suitable for automated analysis (cf. [8], [54], [103], [114]). Cortier et al. [39] overcome this limitation with an alternative definition based on refinement type systems.

Also in the symbolic model, Kremer and Ryan [88] and Backes et al. [9] formalize definitions of *eligibility*. These definitions are not intended to provide assurances if the election authorities are dishonest. For example, the definition of Kremer and Ryan does not detect whether corrupt election authorities insert votes [88, §5.2]. Likewise, the definition of Backes et al. assumes that election authorities are honest [9, §3].

Our definition of election verifiability has been adapted to auction schemes by Quaglia & Smyth [106]. And the definition of election verifiability by Kremer et al. [89] has been adapted to auction [56] and examination [55], [57] schemes. Moreover, McCarthy et al. [97] have shown that auction schemes can be constructed from Helios and JCJ. Thus, our results are applicable beyond voting.

Our definition of election verifiability follows Smyth et al. [89], [111], [120] by deconstructing it into individual, universal, and eligibility verifiability. Other deconstructions of election verifiability are possible. For example, Adida and Neff [6, §2] identify four aspects of verifiability:

- *Cast as intended:* the ballot is cast at the polling station as the voter intended.
- *Recorded as cast:* cast ballots are preserved with integrity through the ballot collection process.
- *Counted as recorded:* recorded ballots are counted correctly.
- *Eligible voter verification:* only eligible voters can cast a ballot in the first place.

Those definitions are not mathematical, so we cannot attempt a precise comparison. Nonetheless, eligibility verifiability and eligible voter verification seem to be addressing similar concerns. Likewise, individual and universal verifiability together seem to be addressing concerns similar to that of recorded as cast and counted as recorded together. Recorded as cast, in our work, reduces to the bulletin board preserving ballots with integrity—a property that we have assumed, because cryptographic election schemes assume it, too. Ways to construct secure bulletin boards have been proposed, e.g., [50], [72], [105], [108]. We postpone a discussion of cast as intended to Section IX.

Privacy properties [53], [83], [93], [94], [115], [117]—such as ballot secrecy, receipt freeness, and coercion resistance—complement verifiability. Chevallier-Mames et al. [34], [35] and Hosp and Vora [75], [76] show an incompatibility result: election schemes cannot unconditionally satisfy privacy and universal verifiability. But weaker versions of these properties can hold simultaneously, as can be witnessed from Theorems 4 and 8 coupled with existing privacy results such as the ballot secrecy proofs for Helios'12 [21, Theorem 3], [20, Theorem 6.12], and the coercion resistance proof for JCJ [83, §5].

Comparison with global verifiability: Küsters et al. [92], [93], [95] present a definition of *global verifiability* that can be used with any kind of protocol, not just electronic voting protocols. To analyze the verifiability of a protocol, users of this definition must themselves formalize *goals*, which are properties required to hold in every run of the protocol. For example, a goal γ_ℓ is presented in a case study [93, §5.2] of global verifiability applied to voting:

> γ_ℓ contains all runs for which there exist choices of the dishonest voters (where a choice is either to abstain or to vote for one of the candidates) such that the result obtained together with the choices made by the honest voters in this run differs only by ℓ votes from the published result (i.e. the result that can be computed from the simple ballots on the bulletin board).

Another goal γ is presented in a case study [95, §6.2] of Helios:

> γ is satisfied in a run if the published result exactly reflects the actual votes of the honest voters in this run and votes of dishonest voters are distributed in some way on the candidates, possibly in a different way than how the dishonest voters actually voted.

These informal statements of goals are appealing, but they do not constitute rigorous mathematical definitions. As Kiayias et al. write, "[global verifiability] has the disadvantage that the set γ remains undetermined and thus the level of verifiability that is offered by the definition hinges on the proper definition of γ which may not be simple" [87, p. 476]. In our own work, we found that formal definitions were quite tricky to get right—for example, which ballots should be counted, how to count them, and how to determine whether that count differed from the published tally. So we shared[36] and discussed[37] our results with Küsters. In response, Küsters et al. updated an online technical report to propose a formalization of goals [90, §5.2]; we look forward to analyzing that formalization when it is formally published.

In an analysis of Helios, Küsters et al. [95] use goal γ to conclude that Helios 2.0 satisfies global verifiability. Yet Bernhard et al. [21] demonstrate a vulnerability against the verifiability of Helios 2.0, and in Appendix D we show that Helios 2.0 does not satisfy Ver-Ext. This seeming discrepancy arises because the analysis in [95] does not formalize all the cryptographic primitives used by Helios, hence the vulnerability goes unnoticed. So another contribution of our own work is to correctly distinguish between unverifiable and verifiable variants of Helios by rigorously analyzing the cryptography used in Helios.

It is natural to ask whether election verifiability can be expressed in terms of global verifiability. We believe it can be. For instance, individual, universal and eligibility verifiability could be expressed, in the informal style of the goals quoted above, as the following goals:

- γ_{IV} is satisfied in a run if voters can uniquely identify their ballots on the bulletin board in this run.
- γ_{UV} is satisfied in a run if the correct tally of votes cast by authorized voters in this run is the same as the tally produced by algorithm Tally.
- γ_{EV} is satisfied in a run if every ballot tallied in this run was created by a voter in possession of a private credential.

More concretely, Cortier et al. [43] formalize a goal that is intended to express our definition of election verifiability with external authentication.[38] They also formalize goals intended to express definitions of election verifiability by Cohen and Fischer [14], [38], Kiayias et al. [87], and Cortier et al. [41].

Küsters et al. [93] argue that deconstructing verifiability into individual and universal verifiability is insufficient to detect certain attacks involving ill-formed ballots. But those attacks leave open the possibility that there do exist notions of individual and universal verifiability that would be sufficient. Indeed, our own definition of universal verifiability rules out attacks based on ill-formed ballots, because *correct-tally* ensures that tallied ballots are well-formed. And Cortier et al. claim that their definitions of individual and universal verifiability also rule out such attacks [39, §1].

One concern that might be raised is whether there still lurk any "gaps" in our decomposition into individual and universal (and eligibility) verifiability. Indeed, there might be. But the definition of global verifiability does not rule out the possibility of gaps, either: any gap in the formal statement of a goal will lead to a vulnerability. That is, if the analyst forgets to include some necessary facet of verifiability when stating the formal goal, then global verifiability will not detect any attacks against that facet. Indeed, Cortier et al. [43, §1] state that some of the goals they formalize have "severe limitations and weaknesses." Global verifiability does not guarantee a lack of gaps.

IX. Concluding Remarks

When we began this work, we were studying the Juels et al. [83] definition of election verifiability. We discovered that the definition fails to detect biasing and collusion attacks. While attempting to improve the Juels et al. definition to rule out those attacks, we discovered that factoring it into individual, universal, and eligibility verifiability led to an elegant decomposition of (mostly) orthogonal properties. We later sought to apply our new definitions to existing electronic voting systems, and Helios [5] and JCJ [83] were natural choices. But they treat authentication differently—Helios outsources authentication, whereas JCJ does not—so we were led to separate our definitions into variants for external and internal authentication. We were at first surprised to discover that JCJ does not satisfy the strong definition of eligibility verifiability. But upon reflection, it became apparent that an adversary who knows the tallier's private key can easily forge ballots that appear to be from eligible voters. Helios-C [41], however, avoids this problem by employing digital signatures.

Our definitions of verifiability have not addressed the issue of voter intent—that is, verification by a human that the ballot submitted by a voter corresponds to the candidate choice the voter intended to make. Adida and Neff call this property "cast as intended" [6]. Many election schemes (e.g., [60], [74], [83], [87]) do not satisfy cast as intended, because the schemes implicitly or explicitly assume that voters can themselves verify the cryptographic operations required to construct ballots. Nevertheless, schemes by Chaum [29], Neff [100], and Benaloh [15], [16] introduce cryptographic mechanisms to verify voter intent. It would be natural to explore strengthening our definitions to address voter intent.

The goal of this research is to enable verifiability of the voting systems we use in real-life, rather than merely trusting them. Research on verifiability can generalize beyond voting to other systems that must guarantee strong forms of integrity. Verifiable voting systems thus have the potential to contribute to the science of security, to democracy, and to broader society.

Acknowledgments

We thank David Bernhard, Jeremy Clark, Véronique Cortier, David Galindo, Stéphane Glondu, Markus Jakobsson, Steve Kremer, Ralf Küsters, Elizabeth Quaglia, Mark Ryan, Susan Thomson, and Poorvi Vora for insightful discussions that have influenced this paper. This work is partly supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC project

36. Ralf Küsters, email communication, 24 June 2014.

37. Ralf Küsters, email communication, October/November 2014.

38. Cortier et al. [43, §8.5 & §10.1] incorrectly claim that our definition of election verifiability admits an election scheme which it should not: the election scheme in which "Vote always [outputs error symbol $\perp$] for some dishonestly generated public key [and Tally behaves normally]." Our definition does indeed admit this scheme, because it is verifiable. Indeed, voters can detect that ballots are not recorded. Cortier et al. [43, §10.1] also incorrectly claim that we trust the bulletin board and assume all voters will run the correct Vote algorithm, we do not (cf. §II-B1 and §II-B2).

CRYSP (259639), by AFOSR grants FA9550-12-1-0334 and FA9550-14-1-0334, by NSF grant 1421373, and by the National Security Agency. This work was performed in part at George Washington University and INRIA.

DEDICATION[39]

Ben Smyth dedicates his contribution to the loving memory of Anne Konishi, 1971 – 2015. What matters most of all is the dash. We had a great time.

He writes for Christina Mai Konishi. Smile like your mother, for good fortune seeks those who smile (*warau kado niwa fuku kitaru*, says the Japanese proverb).

APPENDIX A
CRYPTOGRAPHIC PRIMITIVES

A. Basic definitions

Definition 12 (Negligible function [64])**.** *A function* $\mu : \mathbb{N} \rightarrow \mathbb{R}$ *is* negligible *if for every positive polynomial function* $p(\cdot)$*, there exists an* N*, such that for all* $n > N$*,*

$$\mu(n) < \frac{1}{p(n)}.$$

An event $E(k)$, where k is a security parameter, occurs with *negligible probability* if $\Pr[E(k)] \leq \mu(k)$ for some negligible function μ. The event occurs with *overwhelming probability* if the complement of the event occurs with negligible probability.

Definition 13 (Asymmetric encryption scheme [84])**.** *An* asymmetric encryption scheme *is a tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

- **Gen***, denoted* $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(k)$*, takes a security parameter* k *as input and outputs a key pair* (pk, sk) *and message space* $\mathfrak{m}$*.*
- **Enc***, denoted* $c \leftarrow \mathsf{Enc}(pk, m)$*, takes a public key* pk *and message* $m \in \mathfrak{m}$ *as input, and outputs a ciphertext* c*.*
- **Dec***, denoted* $m \leftarrow \mathsf{Dec}(sk, c)$*, takes a private key* sk*, and ciphertext* c *as input, and outputs a message* m *or error symbol* $\perp$*. We assume* Dec *is deterministic.*

Moreover, the scheme must be correct*: there exists a negligible function* μ*, such that for all security parameters* k *and messages* m*, we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(k); c \leftarrow \mathsf{Enc}(pk, m) : m \in \mathfrak{m} \Rightarrow \mathsf{Dec}(sk, c) = m] > 1 - \mu(k)$*.*

Our definition of asymmetric encryption schemes differs from Katz and Lindell's definition [84, Definition 10.1] in that we formally state the plaintext space.

Definition 14 (Homomorphic encryption)**.** *An asymmetric encryption scheme* $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* homomorphic*, with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$*,*[40] *if there exists a negligible function* μ*, such that for all security parameters* k*, we have the following.*[41] *First, for all messages* m_1 *and* m_2 *we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(k); c_1 \leftarrow \mathsf{Enc}(pk, m_1); c_2 \leftarrow \mathsf{Enc}(pk, m_2) : m_1, m_2 \in \mathfrak{m} \Rightarrow \mathsf{Dec}(sk, c_1 \otimes_{pk} c_2) = \mathsf{Dec}(sk, c_1) \odot_{pk} \mathsf{Dec}(sk, c_2)] > 1 - \mu(k)$*. Secondly, for all messages* m_1 *and* m_2*, and coins* r_1 *and* r_2*, we have* $\Pr[(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(k) : m_1, m_2 \in \mathfrak{m} \Rightarrow \mathsf{Enc}(pk, m_1; r_1) \otimes_{pk} \mathsf{Enc}(pk, m_2; r_2) = \mathsf{Enc}(pk, m_1 \odot_{pk} m_2; r_1 \oplus_{pk} r_2)] > 1 - \mu(k)$*.*

We say Γ *is* additively homomorphic*, respectively* multiplicatively homomorphic*, if for all security parameters* k*, key pairs* pk, sk*, and message spaces* $\mathfrak{m}$*, such that there exists coins* r *and* $(pk, sk, \mathfrak{m}) = \mathsf{Setup}(k)$*, we have* $\odot_{pk}$ *is the addition operator, respectively multiplication operator, in group* $(\mathfrak{m}, \odot_{pk})$*.*

Indistinguishability under chosen-plaintext attack ($\mathsf{IND\text{-}CPA}$) [10], [12], [13], [65], [66] is a standard definition of security for encryption schemes. Intuitively, if an encryption scheme satisfies $\mathsf{IND\text{-}CPA}$, then an adversary without access to a decryption oracle is unable to distinguish ciphertexts. A variant ($\mathsf{IND}\text{-}j\text{-}\mathsf{CPA}$) allows the adversary j adaptive queries to a decryption oracle, where each query is a *parallel decryption query*—i.e., it requests the decryption of a vector of ciphertexts. Hence, $\mathsf{IND\text{-}0\text{-}CPA}$ is equivalent to $\mathsf{IND\text{-}CPA}$.

Definition 15 ($\mathsf{IND}\text{-}j\text{-}\mathsf{CPA}$ [22])**.** *An asymmetric encryption scheme* $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies* $\mathsf{IND}\text{-}j\text{-}\mathsf{CPA}$ *if for all stateful PPT adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, we have* $\mathsf{Succ}(\mathsf{Exp\text{-}CPA}(j, \Gamma, \mathcal{A}, k)) \leq \frac{1}{2} + \mu(k)$*, where* j *is a non-negative integer and the experiment* $\mathsf{Exp\text{-}CPA}$ *is defined as follows:*[42]

$\mathsf{Exp\text{-}CPA}(j, \Gamma, \mathcal{A}, k) =$
$(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(k)$;
$(m_0, m_1) \leftarrow \mathcal{A}(pk, \mathfrak{m})$;
$b \leftarrow_R \{0, 1\}$;
$c \leftarrow \mathsf{Enc}(pk, m_b)$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(c)$;
if $b = b' \wedge m_0, m_1 \in \mathfrak{m} \wedge |m_0| = |m_1|$ **then**
  return 1
else
  return 0

where $\mathcal{A}$ *has access to a decryption oracle* $\mathcal{O}$*, which is defined as follows*[43]*:*

$\mathcal{O}(\mathbf{c}) =$

39. The dedication references Linda Ellis (1996) *The Dash*.

40. Henceforth, we implicitly bind ternary operators—i.e., we write Γ *is a homomorphic asymmetric encryption scheme* as opposed to the more verbose Γ *is a homomorphic asymmetric encryption scheme, with respect to ternary operators* $\odot$*,* $\oplus$*, and* $\otimes$.

41. We write $X \circ_{pk} Y$ for the application of ternary operator $\circ$ to inputs X, Y, and pk. We occasionally abbreviate $X \circ_{pk} Y$ as $X \circ Y$, when pk is clear from the context.

42. Let $x \leftarrow_R S$ denote assignment to x of an element chosen uniformly at random from set S.

43. The oracle in experiment $\mathsf{Exp\text{-}CPA}$ may access parameter j. Henceforth, we continue to allow oracles to access experiment parameters without explicitly mentioning them.

if $j > 0 \wedge \bigwedge_{1 \leq i \leq |\mathbf{c}|} c \neq \mathbf{c}[i]$ **then**
  $j \leftarrow j - 1$;
  return $(\mathsf{Dec}(sk, \mathbf{c}[1]), \ldots, \mathsf{Dec}(sk, \mathbf{c}[|\mathbf{c}|]))$
else
  return $\perp$

Definition 16 (Signature scheme [84])**.** *A* signature scheme *is a tuple* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *of PPT algorithms such that:*

- **Gen***, denoted* $(pk, sk) \leftarrow \mathsf{Gen}(k)$*, takes a security parameter* k *as input and outputs a key pair* (pk, sk)*.*
- **Sign***, denoted* $\sigma \leftarrow \mathsf{Sign}(sk, m)$*, takes a private key* sk *and message* m *as input, and outputs a signature* σ*.*
- **Verify***, denoted* $v \leftarrow \mathsf{Ver}(pk, m, \sigma)$*, takes a public key* pk*, message* m*, and signature* σ *as input, and outputs a bit* v*, which is* 1 *if the signature successfully verifies and* 0 *otherwise. We assume* Ver *is deterministic.*

Moreover, the scheme must be correct*: there exists a negligible function* μ*, such that for all security parameters* k *and messages* m*, we have* $\Pr[(pk, sk) \leftarrow \mathsf{Gen}(k); \sigma \leftarrow \mathsf{Sign}(sk, m); \mathsf{Ver}(pk, m, \sigma) = 1] > 1 - \mu(k)$*.*

Definition 17 (EU-CMA [84])**.** *A signature scheme* $\Gamma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *satisfies* existential unforgeablility under adaptive chosen-message attack (EU-CMA) *if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, we have* $\mathsf{Succ}(\mathsf{Exp\text{-}Sign}(\Gamma, \mathcal{A}, k)) \leq \mu(k)$*, where experiment* $\mathsf{Exp\text{-}Sign}$ *is defined as follows:*

$\mathsf{Exp\text{-}Sign}(\Gamma, \mathcal{A}, k) =$
$(pk, sk) \leftarrow \mathsf{Gen}(k)$;
$Msg \leftarrow \emptyset$;
$(m, \sigma) \leftarrow A^{\mathcal{O}}(pk, k)$;
if $\mathsf{Ver}(pk, m, \sigma) = 1 \wedge m \notin Msg$ **then**
  return 1
else
  return 0

The experiment defines an oracle $\mathcal{O}$*. On invocation* $\mathcal{O}(m)$*, oracle* $\mathcal{O}$ *computes a signature* $\sigma \leftarrow \mathsf{Sign}(sk, m)$*, records that the adversary requested a signature on* m *by updating* Msg *to be* $Msg \cup \{m\}$*, and outputs* σ*.*

Definition 18. *A signature scheme* $\Gamma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *satisfies* strong unforgeability *if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all security parameters* k*, we have* $\mathsf{Succ}(\mathsf{Exp\text{-}StrongSign}(\Gamma, \mathcal{A}, k)) \leq \mu(k)$*, where experiment* $\mathsf{Exp\text{-}StrongSign}$ *is defined as follows:*

$\mathsf{Exp\text{-}StrongSign}(\Gamma, \mathcal{A}, k) =$
$(pk, sk) \leftarrow \mathsf{Gen}(k)$;
$Msg \leftarrow \emptyset$;
$(m, \sigma) \leftarrow A^{\mathcal{O}}(pk, k)$;
if $\mathsf{Ver}(pk, m, \sigma) = 1 \wedge (m, \sigma) \notin Msg$ **then**
  return 1
else
  return 0

The experiment defines an oracle $\mathcal{O}$*. On invocation* $\mathcal{O}(m)$*, oracle* $\mathcal{O}$ *computes a signature* $\sigma \leftarrow \mathsf{Sign}(sk, m)$*, records the request and response* (m, σ) *by updating* Msg *to be* $Msg \cup \{(m, \sigma)\}$*, and outputs* σ*.*

B. Proof systems

A *proof system* (originally known as an *interactive proof system* [67]) is a two-party protocol between a prover and a verifier. The prover convinces the verifier that a string x is in a language L. Here, we assume that there is a *witness relation* R, such that $s \in L$ iff there exists a witness w, such that $(s, w) \in R$. For any $(s, w) \in R$, it must also hold that the length of w is at most polynomial in the length of s. Proof systems ensure that a prover can convince a verifier of any valid claim (*completeness*), and that a verifier cannot be fooled into accepting a false claim (*soundness*).

A *sigma protocol* [51], [71] is a proof system with a particular three-move structure: commit, challenge, respond.

Definition 19 (Sigma protocol)**.** *A* sigma protocol *for a relation* R *is a tuple* $(\mathsf{Comm}, \mathsf{Chal}, \mathsf{Resp}, \mathsf{Verify})$ *of PPT algorithms such that:*

- **Comm***, denoted* $(\mathsf{comm}, t) \leftarrow \mathsf{Comm}(s, w, k)$*, is executed by a* prover*.* Comm *takes a statement* s*, witness* w *and security parameter* k *as input, and outputs a commitment* comm *and some state information* t*.*
- **Chal***, denoted* $\mathsf{chal} \leftarrow \mathsf{Chal}(s, \mathsf{comm}, k)$*, is executed by a* verifier*.* Chal *takes a statement* s*, a commitment* comm *and a security parameter* k *as input, and outputs a string* chal.
- **Resp***, denoted* $\mathsf{resp} \leftarrow \mathsf{Resp}(\mathsf{chal}, t, k)$*, is executed by a prover.* Resp *takes a challenge* chal*, state information* t *and security parameter* k *as input, and outputs a response* resp.
- **Verify***, denoted* $v \leftarrow \mathsf{Verify}(s, (\mathsf{comm}, \mathsf{chal}, \mathsf{resp}), k)$ *is executed by a verifier.* Verify *takes a statement* s*, a transcript* $(\mathsf{comm}, \mathsf{chal}, \mathsf{resp})$ *and a security parameter* k *as input, and outputs a bit* v*, which is* 1 *if the transcript successfully verifies and* 0 *otherwise. We assume* Verify *is deterministic.*

Moreover, the sigma protocol must be complete*: there exists a negligible function* μ*, such that for all statements and witnesses* $(s, w) \in R$ *and security parameters* k*, we have* $\Pr[(\mathsf{comm}, t) \leftarrow \mathsf{Comm}(s, w, k); \mathsf{chal} \leftarrow \mathsf{Chal}(s, \mathsf{comm}, k); \mathsf{resp} \leftarrow \mathsf{Resp}(\mathsf{chal}, t, k) : \mathsf{Verify}(s, (\mathsf{comm}, \mathsf{chal}, \mathsf{resp}), k) = 1] > 1 - \mu(k)$*.*

Some sigma protocols ensure *special soundness* and *special honest-verifier zero-knowledge*. We will make use of a result by Bernhard et al. that requires these properties, but we will not need the details of those definitions in our proofs, so we omit them here; see Bernhard et al. [21] for a formalization.

Definition 20. *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a homomorphic asymmetric encryption scheme and* Σ *be a sigma protocol for a* *binary relation R.*[44]

- Σ proves correct key construction *if*

$$((k, pk, \mathfrak{m}), (sk, r)) \in R \Leftrightarrow (pk, sk, \mathfrak{m}) = \mathsf{Gen}(k; r)$$

Further, suppose that $(pk, sk, \mathfrak{m})$ is the output of $\mathsf{Gen}(k; r)$, for some security parameter k and coins r.

- Σ proves plaintext knowledge in a subspace *if*

$$((pk, c, \mathfrak{m}'), (m, r)) \in R \\ \Leftrightarrow c = \mathsf{Enc}(pk, m; r) \wedge m \in \mathfrak{m}' \wedge \mathfrak{m}' \subseteq \mathfrak{m}.$$

- Σ proves conjunctive plaintext knowledge *if*

$$((pk, c_1, \ldots, c_k), (m_1, r_1, \ldots, m_k, r_k)) \in R \\ \Leftrightarrow \bigwedge_{1 \le i \le k} c_i = \mathsf{Enc}(pk, m_i; r_i) \wedge m_i \in \mathfrak{m}.$$

- Σ proves correct reencryption *if*

$$((pk, \mathbf{c}, c), (i, r)) \in R \\ \Leftrightarrow c = \mathbf{c}[i] \otimes \mathsf{Enc}(pk, \mathfrak{e}; r) \wedge 1 \le i \le |\mathbf{c}|$$

  where $\mathbf{c}$ is a vector of ciphertexts encrypted under pk, and where $\mathfrak{e}$ is an identity element of the encryption scheme's message space with respect to $\odot$.

- Σ *is a* plaintext equivalence test *(PET) if*

$$((pk, c, c', i), sk) \in R \\ \Leftrightarrow ((i = 0 \wedge \mathsf{Dec}(sk, c) \neq \mathsf{Dec}(sk, c')) \\ \vee (i = 1 \wedge \mathsf{Dec}(sk, c) = \mathsf{Dec}(sk, c'))) \\ \wedge \mathsf{Dec}(sk, c) \neq \bot \wedge \mathsf{Dec}(sk, c') \neq \bot.$$

- Σ *is a* mixnet *if*

$$((pk, \mathbf{c}, \mathbf{c}'), (\mathbf{r}, \chi)) \in R \\ \Leftrightarrow \bigwedge_{1 \le i \le |\mathbf{c}|} \mathbf{c}'[\chi(i)] = \mathbf{c}[i] \otimes \mathsf{Enc}(pk, \mathfrak{e}; \mathbf{r}[i]) \\ \wedge |\mathbf{c}| = |\mathbf{c}'| = |\mathbf{r}|$$

  where $\mathbf{c}$ and $\mathbf{c}'$ are both vectors of ciphertexts encrypted under pk, and χ is a permutation on $\{1, \ldots, |\mathbf{c}|\}$, and $\mathfrak{e}$ is an identity element of the encryption scheme's message space with respect to $\odot$.

- Σ proves correct decryption *if*

$$((pk, c, m), sk) \in R \Leftrightarrow m = \mathsf{Dec}(sk, c).$$

C. Non-interactive proof systems

A proof system is *non-interactive* if a single message is sent from the prover to the verifier.

Definition 21 (Non-interactive proof system)**.** *A* non-interactive proof system *for a relation R is a tuple of PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *such that:*

- **Prove***, denoted $\sigma \leftarrow \mathsf{Prove}(s, w, k)$, is executed by a prover to prove $(s, w) \in R$.*
- **Verify***, denoted $v \leftarrow \mathsf{Verify}(s, \sigma, k)$, is executed by anyone to check the validity of a proof. We assume* Verify *is deterministic.*

Moreover, the system must be complete*: there exists a negligible function μ, such that for all statement and witnesses $(s, w) \in R$ and security parameters k, we have $\Pr[\sigma \leftarrow \mathsf{Prove}(s, w, k) : \mathsf{Verify}(s, \sigma, k) = 1] > 1 - \mu(k)$.*

We can derive non-interactive proof systems from sigma protocols using the *Fiat-Shamir transformation* [59], which replaces the verifier's challenge with a hash of the prover's commitment, concatenated with the prover's statement.

Definition 22 (Fiat-Shamir transformation [59])**.** *Given a sigma protocol $\Sigma = (\mathsf{Comm}, \mathsf{Chal}, \mathsf{Resp}, \mathsf{Verify}_\Sigma)$ for relation R and a hash function $\mathcal{H}$, the* Fiat-Shamir transformation*, denoted $\mathsf{FS}(\Sigma, \mathcal{H})$, is the tuple $(\mathsf{Prove}, \mathsf{Verify})$ of algorithms, defined as follows:*

$\mathsf{Prove}(s, w, k) =$
$(\mathsf{comm}, t) \leftarrow \mathsf{Comm}(s, w, k)$;
$\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s)$;
$\mathsf{resp} \leftarrow \mathsf{Resp}(\mathsf{chal}, t, k)$;
return $(\mathsf{comm}, \mathsf{resp})$

$\mathsf{Verify}(s, (\mathsf{comm}, \mathsf{resp}), k) =$
$\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s)$;
return $\mathsf{Verify}_\Sigma(s, (\mathsf{comm}, \mathsf{chal}, \mathsf{resp}), k)$

It is straightforward to check that FS produces non-interactive proof systems. In particular, given sigma protocol Σ for relation R, and a hash function $\mathcal{H}$, we have $\mathsf{FS}(\Sigma, \mathcal{H})$ is a non-interactive proof system for relation R.

Some applications of the Fiat-Shamir transformation produce non-interactive proof systems satisfying *zero-knowledge*: anything a verifier can derive about a witness can be derived without interaction with a prover—that is, the prover can be simulated by a PPT algorithm called a *simulator*. We will not need the details of zero-knowledge in our proofs, so we omit them here; see Bernhard et al. [21] or Quaglia & Smyth [106] for formalizations.

In addition, some applications of the Fiat-Shamir transformation produce non-interactive proof systems satisfying *simulation sound extractability*: an *extractor* can recover witnesses from proofs by *rewinding* the prover, as discussed below. (We use extractors in our proofs of theorems, to obtain witnesses from proofs.) We define simulation sound extractability in the *random oracle model* [11]. A random oracle can be *programmed* or *patched*. We will not need the details of how patching works in our proofs, so we omit them here; see Bernhard et al. [21] for a formalization.

Definition 23 (Simulation sound extractability [21], [69])**.** *Suppose that Σ is a sigma protocol for relation R, that $\mathcal{H}$ is a random oracle, and that $(\mathsf{Prove}, \mathsf{Verify})$ is a non-interactive proof system, such that $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}, \mathsf{Verify})$. Further suppose $\mathcal{S}$ is a simulator for $(\mathsf{Prove}, \mathsf{Verify})$ and $\mathcal{H}$ can be*

44. Given a binary relation R, we write $((s_1, \ldots, s_l), (w_1, \ldots, w_k)) \in R \Leftrightarrow P(s_1, \ldots, s_l, w_1, \ldots, w_k)$ for $(s, w) \in R \Leftrightarrow P(s_1, \ldots, s_l, w_1, \ldots, w_k) \wedge s = (s_1, \ldots, s_l) \wedge w = (w_1, \ldots, w_k)$, hence, R is only defined over pairs of vectors of lengths l and k.

patched by $\mathcal{S}$. Proof system (Prove, Verify) *satisfies* simulation sound extractability *if there exists a PPT algorithm $\mathcal{K}$, such that for all PPT adversaries $\mathcal{A}$ and coins r, there exists a negligible function μ, such that for all security parameters k, we have*[45]

$$\begin{aligned}\Pr[\mathbf{P} \leftarrow (); \mathbf{Q} \leftarrow \mathcal{A}^{\mathcal{H},\mathcal{P}}(\text{—}; r); \mathbf{W} \leftarrow \mathcal{K}^{\mathcal{A}'}(\mathbf{H}, \mathbf{P}, \mathbf{Q}) :\\ |\mathbf{Q}| \neq |\mathbf{W}| \vee \exists j \in \{1, \ldots, |\mathbf{Q}|\} \,.\, (\mathbf{Q}[j][1], \mathbf{W}[j]) \notin R \wedge\\ \forall (s, \sigma) \in \mathbf{Q}, (t, \tau) \in \mathbf{P} \,.\, \mathsf{Verify}(s, \sigma, k) = 1 \wedge \sigma \neq \tau] \leq \mu(k)\end{aligned}$$

where $\mathcal{A}(\text{—}; r)$ denotes running adversary $\mathcal{A}$ with an empty input and random coins r, where $\mathbf{H}$ is a transcript of the random oracle's input and output, and where oracles $\mathcal{A}'$ and $\mathcal{P}$ are defined below:

- $\mathcal{A}'()$. *Computes $\mathbf{Q}' \leftarrow \mathcal{A}(\text{—}; r)$, forwarding any of $\mathcal{A}$'s oracle calls to $\mathcal{K}$, and outputs $\mathbf{Q}'$. By running $\mathcal{A}(\text{—}; r)$, $\mathcal{K}$ is rewinding the adversary.*
- $\mathcal{P}(s)$. *Computes $\sigma \leftarrow \mathcal{S}(s); \mathbf{P} \leftarrow (\mathbf{P}[1], \ldots, \mathbf{P}[|\mathbf{P}|], (s, \sigma))$ and outputs σ.*

Algorithm $\mathcal{K}$ is an extractor *for* (Prove, Verify).

Our definition of simulation sound extractability in the random oracle model is an analogue of Groth's definition in the common reference string model [69, §2]. (See Bernhard et al. [21, §1] for a detailed comparison.) Our presentation of simulation sound extractability differs from the presentation by Bernhard et al. [21] by formalizing some of the details.

Bernhard et al. [21] show that non-interactive proof systems derived using the Fiat-Shamir transformation satisfy zero-knowledge and simulation sound extractability:

Theorem 12 (from [21])**.** *Let Σ be a sigma protocol for relation R, and let $\mathcal{H}$ be a random oracle. If Σ satisfies special soundness and special honest verifier zero-knowledge, then $\mathsf{FS}(\Sigma, \mathcal{H})$ satisfies zero-knowledge and simulation sound extractability.*

The Fiat-Shamir transformation can be generalized to include an optional string m in the hashes produced by functions Prove and Verify. We write $\mathsf{Prove}(s, w, m, k)$ and $\mathsf{Verify}(s, (\mathsf{comm}, \mathsf{resp}), m, k)$ for invocations of Prove and Verify which include an optional string. When m is provided, it is included in the hashes in both algorithms. That is, given $\mathsf{FS}(\Sigma, \mathcal{H}) = (\mathsf{Prove}, \mathsf{Verify})$, the hashes are computed as follows in both algorithms: $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm}, s, m)$. Theorem 12 can be extended to this generalization.

Appendix B
Variants of Exp-IV

Our individual verifiability experiment with external authentication (§II-B1) can be equivalently formulated as an experiment that challenges $\mathcal{A}$ to predict the output of Vote:

$\mathsf{Exp\text{-}IV\text{-}Ext}'(\Pi, \mathcal{A}, k) =$

```
(PK_T, n_C, β, b) ← A(k);
b' ← Vote(PK_T, n_C, β, k);
if b = b' ∧ b' ≠ ⊥ then
    return 1
else
    return 0
```

Proposition 13. *Given an election scheme Π, we have*

$$\begin{aligned}&\forall \mathcal{A}\ \exists \mu\ \forall k \,.\, \mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k)) \leq \mu(k)\\ &\Leftrightarrow \forall \mathcal{A}'\ \exists \mu'\ \forall k' \,.\, \mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Ext}'(\Pi, \mathcal{A}', k')) \leq \mu'(k'),\end{aligned}$$

where $\mathcal{A}$ and $\mathcal{A}'$ are PPT adversaries, μ and μ' are negligible functions, and k and k' are security parameters.

Intuitively, if $\mathcal{A}$ can predict the output of Vote, then $\mathcal{A}$ can use that prediction to generate a collision. And if $\mathcal{A}$ can generate collisions, then $\mathcal{A}$ can use them to predict outputs.

Proof. For the forward implication, suppose $\mathcal{A}'$ is a PPT adversary such that $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Ext}'(\Pi, \mathcal{A}', k')) > \frac{1}{p(k')}$ for some polynomial function p and security parameter k'. We construct an adversary $\mathcal{A}$ against Exp-IV-Ext. On input k', adversary $\mathcal{A}$ computes $(PK_{\mathcal{T}}, n_C, \beta, b) \leftarrow \mathcal{A}'(k')$ and outputs $(PK_{\mathcal{T}}, n_C, \beta, \beta)$. Since $\mathcal{A}'$ wins $\mathsf{Exp\text{-}IV\text{-}Ext}'$ with non-negligible probability, we have

$$\Pr[b' \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k') : b = b' \wedge b \neq \bot] > \frac{1}{p(k')}.$$

Moreover, since calls to algorithm Vote are independent, we have

$$\begin{aligned}\Pr[b_1 \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k');&\\ b_2 \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k')&\\ : b_1 = b \wedge b_2 = b \wedge b_1 \neq \bot \wedge b_2 \neq \bot] &> \frac{1}{p(k')^2}.\end{aligned}$$

It follows that $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k')) > \frac{1}{p(k')^2}$.

For the reverse implication, suppose $\mathcal{A}$ is a PPT adversary such that $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Ext}(\Pi, \mathcal{A}, k)) > \frac{1}{p(k)}$ for some polynomial function p and security parameter k. We construct an adversary $\mathcal{A}'$ against $\mathsf{Exp\text{-}IV\text{-}Ext}'$. On input k, adversary $\mathcal{A}'$ computes $(PK_{\mathcal{T}}, n_C, \beta_1, \beta_2) \leftarrow \mathcal{A}(k); b_1 \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta_1, k)$ and outputs $(PK_{\mathcal{T}}, n_C, \beta_2, b_1)$. Since $\mathcal{A}$ wins Exp-IV-Ext with probability no less than $\frac{1}{p(k)}$, we have

$$\Pr[b_2 \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta_2, k) : b_1 = b_2 \wedge b_1 \neq \bot] > \frac{1}{p(k)}.$$

It follows that $\mathsf{Succ}(\mathsf{Exp\text{-}IV\text{-}Int}'(\Pi, \mathcal{A}', k)) > \frac{1}{p(k)}$. □

Our individual verifiability experiment with internal authentication (§IV-B1) can also be reformulated as an experiment that challenges $\mathcal{A}$ to predict the output of Vote algorithms:

45. We extend set membership notation to vectors: we write $x \in \mathbf{x}$ if x is an element of the set $\{\mathbf{x}[i] : 1 \leq i \leq |\mathbf{x}|\}$

$\mathsf{Exp\text{-}IV\text{-}Int}'(\Pi, \mathcal{A}, k) =$

```
1  (PK_T, n_V) ← A(k);
2  for 1 ≤ i ≤ n_V do (pk_i, sk_i) ← Register(PK_T, k)
3  L ← {pk_1, ..., pk_{n_V}};
4  Crpt ← ∅;
5  (n_C, β, i, b) ← A^C(L);
6  b' ← Vote(sk_i, PK_T, n_C, β, k);
7  if b = b' ∧ b' ≠ ⊥ ∧ sk_i ∉ Crpt then
8  |  return 1
9  else
10 |  return 0
```

Similarly to Section IV-B1, the adversary is given access to oracle C and the voter index output on line 5 must be legal with respect to n_V.

Experiment Exp-IV-Int$'$ is strictly stronger than our original experiment Exp-IV-Int, since predicting the output of Vote does not imply the existence of collisions, whereas collisions can be used to predict the output of Vote. For instance, consider the following variant of Nonce (Definition 5):

Definition 24. *Election scheme* Nonce$'$ *is defined as follows:*

- Setup(k) *outputs* $(\perp, \perp, \infty, \infty)$.
- Register$(PK_{\mathcal{T}}, k)$ *computes* $r \in \mathbb{Z}_{2^k}$ *and outputs* (r, r).
- Vote$(r, PK_{\mathcal{T}}, n_C, \beta, k)$ *outputs* (r, β).
- Tally$(SK_{\mathcal{T}}, BB, L, n_C, k)$ *computes a vector* $\mathbf{X}$ *of length* n_C*, such that* $\mathbf{X}$ *is a tally of the votes on* BB *for which the nonce is in* L*, and outputs* $(\mathbf{X}, \perp)$.
- Verify$(PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, P, k)$ *outputs* 1 *if* $(\mathbf{X}, P) =$ Tally$(\perp, \perp, BB, L, n_C, k)$ *and* 0 *otherwise.*

Intuitively, an adversary can predict the output of Vote, because the algorithm is deterministic and the electoral roll lists private credentials. However, the Register algorithm ensures that voters' credentials are distinct with overwhelming probability, hence, instantiations of the Vote algorithm with distinct voter credentials will never collide.

Proposition 14. *Given an election scheme* Π*, PPT adversary* $\mathcal{A}$*, negligible function* μ*, and security parameter* k*, if* Succ(Exp-IV-Int$'(\Pi, \mathcal{A}, k)) \leq \mu(k)$*, then there exists a PPT adversary* $\mathcal{B}$ *such that* Succ(Exp-IV-Int$(\Pi, \mathcal{B}, k)) \leq \mu(k)$.

The proof of Proposition 14 is similar to the reverse implication proof of Proposition 13.

APPENDIX C
GENERALIZED HELIOS SCHEME

We formalize a generic construction for Helios-like election schemes (Figure 1). Our construction is parameterized on the choice of homomorphic encryption scheme and sigma protocols.

Setup generates the tallier's key pair. The public key includes a non-interactive proof that the key pair is correctly constructed. Vote takes a choice $\beta \in \{1, \ldots, n_C\}$ and outputs ciphertexts $c_1, \ldots, c_{n_C - 1}$ such that if $\beta < n_C$, then ciphertext c_β contains plaintext 1 and the remaining ciphertexts contain plaintext 0, otherwise, all ciphertexts contain plaintext 0. Vote also outputs proofs $\sigma_1, ..., \sigma_{n_C}$ so that this can be verified, in particular, proof σ_j demonstrates that the ciphertext c_j contains 0 or 1 for all $1 \leq j \leq n_C - 1$, and the proof σ_{n_C} demonstrates that the homomorphic combination of ciphertexts $c_1 \otimes \cdots \otimes c_{n_C}$ contains 0 or 1 (i.e., the voter's ballot contains a vote for exactly one candidate). Tally homomorphically combines ciphertexts representing votes for a particular candidate and decrypts the homomorphic combinations. The number of votes for a candidate $\beta \in \{1, \ldots, n_C - 1\}$ is simply the homomorphic combination of the ballots for that candidate; the number of votes for candidate n_C is equal to the number of votes for all other candidates subtracted from the total number of valid ballots on the bulletin board. Verify checks that each of the above steps has been performed correctly.

Lemmata 15–17 demonstrate that generalized Helios is a construction for election schemes.

Lemma 15. Helios$(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ *satisfies Correctness, where* Γ*,* Σ_1*,* Σ_2*,* Σ_3 *and* $\mathcal{H}$ *satisfy the preconditions of Figure 1.*

The proof of Lemma 15 is similar to the proof of Proposition 21.

Lemma 16. *Suppose* Γ*,* Σ_1*,* Σ_2*,* Σ_3 *and* $\mathcal{H}$ *satisfy the preconditions of Figure 1. Further suppose that* Σ_2 *satisfies special soundness and special honest verifier zero-knowledge, and* $\mathcal{H}$ *is a random oracle. We have* Helios$(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ *satisfies Completeness.*

Proof. Let Helios$(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ = (Setup, Vote, Tally, Verify), FS$(\Sigma_1, \mathcal{H})$ = (ProveKey, VerKey), FS$(\Sigma_2, \mathcal{H})$ = (ProveCiph, VerCiph), and FS$(\Sigma_3, \mathcal{H})$ = (ProveDec, VerDec). Suppose k is a security parameter, BB is a bulletin board, and n_C is an integer. Further suppose $(PK_{\mathcal{T}}, SK_{\mathcal{T}})$ is a key pair, m_B and m_C are integers, and $(\mathbf{X}, P)$ is a tally, such that $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow$ Setup(k) and $(\mathbf{X}, P) \leftarrow$ Tally$(SK_{\mathcal{T}}, BB, n_C, k)$. Moreover, suppose $|BB| \leq m_B$. We focus on the case $n_C > 1$; the case $n_C = 1$ is similar. By definition of Setup, there exist coins s such that $(pk, sk, \mathfrak{m}) =$ Gen$(k; s)$, $PK_{\mathcal{T}} \leftarrow (pk, \mathfrak{m}, \rho)$, $SK_{\mathcal{T}} \leftarrow (pk, sk)$ and m_B is the largest integer such that $\{0, \ldots, m_B\} \subseteq \mathfrak{m}$, where ρ is an output of ProveKey$((k, pk, \mathfrak{m}), (sk, s), k)$. By definition of Tally, we have $\mathbf{X}$ is a vector of length n_C and P is a vector of length $n_C - 1$. It follows that Verify can successfully parse $\mathbf{X}$, P, and $PK_{\mathcal{T}}$. Moreover, by the completeness of (ProveKey, VerKey), we have VerKey$((k, pk, \mathfrak{m}), \rho, k) = 1$ with overwhelming probability. Let $\{b_1, \ldots, b_\ell\}$ be the largest subset of BB satisfying the conditions given by the tally algorithm. If $\{b_1, \ldots, b_\ell\} = \emptyset$, then $\mathbf{X}$ is a zero-filled vector and Verify outputs 1, concluding our proof, otherwise, we proceed as follows. Since $\{b_1, \ldots, b_\ell\}$ is a subset of BB, we have $\ell \leq m_B$. By definition of Tally, we have for all $1 \leq i \leq \ell$ that $\bigwedge_{j=1}^{n_C - 1}$ VerCiph$((pk, b_i[j], \{0, 1\}), b_i[j + n_C - 1], j, k) = 1$. By Theorem 12, we have (ProveCiph, VerCiph) satisfies simulation sound extractability, hence, for all $1 \leq i \leq \ell$ and all $1 \leq j \leq n_C - 1$ we have $b_i[j]$ is a ciphertext with overwhelming probability. It follows for all $1 \leq j \leq n_C - 1$

Fig. 1 Generalized Helios

Suppose $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an additively homomorphic asymmetric encryption scheme with a message space that, for sufficiently large security parameters, includes $\{0,1\}$, Σ_1 proves correct key construction, Σ_2 proves plaintext knowledge in a subspace, Σ_3 proves correct decryption, and $\mathcal{H}$ is a hash function. Let $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, and $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$. We define *generalized Helios* $\mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ as follows.

- $\mathsf{Setup}(k)$. Select coins s, compute $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(k; s); \rho \leftarrow \mathsf{ProveKey}((k, pk, \mathfrak{m}), (sk, s), k); PK_{\mathcal{T}} \leftarrow (pk, \mathfrak{m}, \rho); SK_{\mathcal{T}} \leftarrow (pk, sk)$, let m be the largest integer such that $\{0, \dots, m\} \subseteq \mathfrak{m}$, and output $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m, m)$.
- $\mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$. Parse $PK_{\mathcal{T}}$ as a vector $(pk, \mathfrak{m}, \rho)$. Output $\perp$ if parsing fails or $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) \neq 1 \vee \beta \notin \{1, \dots, n_C\}$. Select coins $r_1, \dots, r_{n_C-1}$ and compute:

  for $1 \le j \le n_C - 1$ **do**
      if $j = \beta$ **then** $m_j \leftarrow 1$ **else** $m_j \leftarrow 0$
      $c_j \leftarrow \mathsf{Enc}(pk, m_j; r_j)$;
      $\sigma_j \leftarrow \mathsf{ProveCiph}((pk, c_j, \{0,1\}), (m_j, r_j), j, k)$
  $c \leftarrow c_1 \otimes \dots \otimes c_{n_C-1}$;
  $m \leftarrow m_1 \odot \dots \odot m_{n_C-1}$;
  $r \leftarrow r_1 \oplus \dots \oplus r_{n_C-1}$;
  $\sigma_{n_C} \leftarrow \mathsf{ProveCiph}((pk, c, \{0,1\}), (m, r), n_C, k)$

  Output ballot $(c_1, \dots, c_{n_C-1}, \sigma_1, \dots, \sigma_{n_C})$.
- $\mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, k)$. Initialize vectors $\mathbf{X}$ of length n_C and $\mathbf{P}$ of length $n_C - 1$. Compute **for** $1 \le j \le n_C$ **do** $\mathbf{X}[j] \leftarrow 0$. Parse $SK_{\mathcal{T}}$ as a vector (pk, sk). Output $(\mathbf{X}, \mathbf{P})$ if parsing fails. Let $\{b_1, \dots, b_\ell\}$ be the largest subset of BB such that $b_1 < \dots < b_\ell$ and for all $1 \le i \le \ell$ we have b_i is a vector of length $2 \cdot n_C - 1$ and $\bigwedge_{j=1}^{n_C-1} \mathsf{VerCiph}((pk, b_i[j], \{0,1\}), b_i[j + n_C - 1], j, k) = 1 \wedge \mathsf{VerCiph}((pk, b_i[1] \otimes \dots \otimes b_i[n_C - 1], \{0,1\}), b_i[2 \cdot n_C - 1], n_C, k) = 1$. If $\{b_1, \dots, b_\ell\} = \emptyset$, then output $(\mathbf{X}, \mathbf{P})$, otherwise, compute:

  for $1 \le j \le n_C - 1$ **do**
      $c \leftarrow b_1[j] \otimes \dots \otimes b_\ell[j]$;
      $\mathbf{X}[j] \leftarrow \mathsf{Dec}(sk, c)$;
      $\mathbf{P}[j] \leftarrow \mathsf{ProveDec}((pk, c, \mathbf{X}[j]), sk, k)$
  $\mathbf{X}[n_C] \leftarrow \ell - \sum_{j=1}^{n_C-1} \mathbf{X}[j]$;

  Output $(\mathbf{X}, \mathbf{P})$.
- $\mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, \mathbf{P}, k)$. Parse $\mathbf{X}$ as a vector of length n_C, parse $\mathbf{P}$ as a vector of length $n_C - 1$, parse $PK_{\mathcal{T}}$ as a vector $(pk, \mathfrak{m}, \rho)$. Output 0 if parsing fails or $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) \neq 1$. Let $\{b_1, \dots, b_\ell\}$ be the largest subset of BB satisfying the conditions given by the tally algorithm and let m_B be the largest integer such that $\{0, \dots, m_B\} \subseteq \mathfrak{m}$. If $\{b_1, \dots, b_\ell\} = \emptyset \wedge \bigwedge_{j=1}^{n_C} \mathbf{X}[j] = 0$ or $\bigwedge_{j=1}^{n_C-1} \mathsf{VerDec}((pk, b_1[j] \otimes \dots \otimes b_\ell[j], \mathbf{X}[j]), \mathbf{P}[j], k) = 1 \wedge \mathbf{X}[n_C] = \ell - \sum_{j=1}^{n_C-1} \mathbf{X}[j] \wedge 1 \le \ell \le m_B$, then output 1, otherwise, output 0.

The above algorithms assume $n_C > 1$ and we define special cases of Vote, Tally and Verify when $n_C = 1$:

- $\mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$. Parse $PK_{\mathcal{T}}$ as a vector $(pk, \mathfrak{m}, \rho)$. Output $\perp$ if parsing fails or $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) \neq 1 \vee \beta \neq 1$. Select coins r, compute $m \leftarrow 1; c \leftarrow \mathsf{Enc}(pk, m; r); \sigma \leftarrow \mathsf{ProveCiph}((pk, c, \{0,1\}), (m, r), k)$, and output ballot (c, σ).
- $\mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, k)$. Initialize $\mathbf{X}$ and $\mathbf{P}$ as vectors of length 1. Compute $\mathbf{X}[1] \leftarrow 0$. Parse $SK_{\mathcal{T}}$ as a vector (pk, sk). Output $(\mathbf{X}, \mathbf{P})$ if parsing fails. Let $\{b_1, \dots, b_\ell\}$ be the largest subset of BB such that for all $1 \le i \le \ell$ we have b_i is a vector of length 2 and $\mathsf{VerCiph}((pk, b_i[1], \{0,1\}), b_i[2], k) = 1$. If $\{b_1, \dots, b_\ell\} = \emptyset$, then output $(\mathbf{X}, \mathbf{P})$. Otherwise, compute $c \leftarrow b_1[1] \otimes \dots \otimes b_\ell[1]; \mathbf{X}[1] \leftarrow \mathsf{Dec}(sk, c); \mathbf{P}[1] \leftarrow \mathsf{ProveDec}((pk, c, \mathbf{X}[1]), sk, k)$ and output $(\mathbf{X}, \mathbf{P})$.
- $\mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, \mathbf{P}, k)$. Parse $\mathbf{X}$ and $\mathbf{P}$ as vectors of length 1, and parse $PK_{\mathcal{T}}$ as a vector $(pk, \mathfrak{m}, \rho)$. Output 0 if parsing fails or $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) \neq 1$. Let $\{b_1, \dots, b_\ell\}$ be the largest subset of BB satisfying the conditions given by the tally algorithm and let m_B be the largest integer such that $\{0, \dots, m_B\} \subseteq \mathfrak{m}$. If $\{b_1, \dots, b_\ell\} = \emptyset \wedge \mathbf{X}[1] = 0$ or $\mathsf{VerDec}((pk, b_1[1] \otimes \dots \otimes b_\ell[1], \mathbf{X}[1]), \mathbf{P}[1], k) = 1 \wedge 1 \le \ell \le m_B$, then output 1, otherwise, output 0.

---

that $b_1[j] \otimes \dots \otimes b_\ell[j]$ is a ciphertext with overwhelming probability. By definition of Tally and the completeness of $(\mathsf{ProveDec}, \mathsf{VerDec})$, we have $\bigwedge_{j=1}^{n_C-1} \mathsf{VerDec}((pk, b_1[j] \otimes \dots \otimes b_\ell[j], \mathbf{X}[j]), P[j], k) = 1 \wedge \mathbf{X}[n_C] = \ell - \sum_{j=1}^{n_C-1} \mathbf{X}[j]$ with overwhelming probability, hence, Verify outputs 1 with overwhelming probability, concluding our proof. □

Definition 25 (Collision-free)**.** *Suppose* $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an asymmetric encryption scheme,* Σ_1 *proves correct key construction,* $\mathcal{H}$ *is a hash function, and* $\mathfrak{m}$ *is a message space. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$*. If for all security* *parameters k, public keys pk, proofs ρ, messages $m_1, m_2 \in \mathfrak{m}$, and coins r_1 and r_2, we have*

$$\begin{aligned}\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) = 1 \wedge (m_1 \neq m_2 \vee r_1 \neq r_2)\\ \Rightarrow \mathsf{Enc}(pk, m_1; r_1) \neq \mathsf{Enc}(pk, m_2; r_2)\end{aligned}$$

Then we say Γ is collision-free for $\mathfrak{m}$.

Lemma 17. *Suppose Γ, Σ_1, Σ_2, Σ_3 and $\mathcal{H}$ satisfy the preconditions of Figure 1. Further suppose Γ is collision-free for $\{0,1\}$. We have $\mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ satisfies Injectivity.*

Proof. Let $\mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$, $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$. Suppose k is a security parameter, $PK_{\mathcal{T}}$ is a public key, n_C is an integer, and β and β' are choices such that $\beta \neq \beta'$. Further suppose b and b' are ballots such that $b \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$, $b' \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta', k)$, $b \neq \perp$, and $b' \neq \perp$. By definition of Vote, we have $PK_{\mathcal{T}}$ is a vector $(pk, \mathfrak{m}, \rho)$ and $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) = 1$. Moreover, there exist coins r and r' such that

$$b[1] = \mathsf{Enc}(pk, m; r), \text{ where } m = \begin{cases} 1 & \text{if } \beta = 1 \\ 0 & \text{otherwise} \end{cases}$$

and

$$b'[1] = \mathsf{Enc}(pk, m'; r'), \text{ where } m' = \begin{cases} 1 & \text{if } \beta' = 1 \\ 0 & \text{otherwise} \end{cases}$$

Since $\beta \neq \beta'$, we have $m \neq m'$. Furthermore, since Γ if collision-free for $\{0,1\}$, we have $b[1] \neq b'[1]$ and, therefore, $b \neq b'$. $\square$

Appendix D
Proof: Helios 2.0 is not verifiable

Bernhard et al. [21] demonstrate that Helios 2.0 [5] is not verifiable and we show that Helios 2.0 does not satisfy Ver-Ext.

Definition 26 (Weak Fiat-Shamir transformation [21])**.** *The* weak Fiat-Shamir transformation *is a function* wFS *that is identical to* FS*, except that it excludes statement s in the hashes computed by* Prove *and* Verify*, as follows:* $\mathsf{chal} \leftarrow \mathcal{H}(\mathsf{comm})$.

Definition 27 (Helios 2.0)**.** *Let $\widehat{\mathsf{Helios}}$ be* Helios *after replacing all instances of the Fiat-Shamir transformation with the weak Fiat-Shamir transformation and excluding the (optional) messages input to* ProveCiph*—i.e.,* $\widehat{\mathsf{ProveCiph}}$ *should be used as a ternary function.* Helios 2.0 *is* $\widehat{\mathsf{Helios}}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$*, where Γ, Σ_1, Σ_2, Σ_3 and $\mathcal{H}$ are given in Definition 28.*

Proposition 18. *Helios 2.0 does not satisfy* Ver-Ext.

Our proof of Proposition 18 formalizes the attack by Bernhard et al. [21, §3] in the context of our universal verifiability experiment.

Proof. Let Vote and Tally be the vote and tallying algorithms defined by Helios 2.0. Moreover, let $\mathsf{wFS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{wFS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$ and $\mathsf{wFS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$. We construct an adversary $\mathcal{A}$ (Figure 2) against the universal verifiability experiment.

Fig. 2 Adversary against Helios 2.0

Given a security parameter k as input, $\mathcal{A}$ computes primes p and 1 such that $p = 2 \cdot q + 1$ and q is of length k. $\mathcal{A}$ also computes a generator g of the multiplicative group $\mathbb{Z}_p^*$. Let $n_C \leftarrow 2$ and $\mathfrak{m} \leftarrow \mathbb{N}_{q-1}$, moreover, let $m > 1$ be an element of $\mathfrak{m}$. The adversary proceeds as follows:

`%coins`
$(a_0, b_0, a_1, b_1) \leftarrow_R \mathbb{Z}_q^4$;
`%witnesses`
$A_0 \leftarrow g^{a_0} \pmod p$;
$B_0 \leftarrow g^{b_0} \pmod p$;
$A_1 \leftarrow g^{a_1} \pmod p$;
$B_1 \leftarrow g^{b_1} \pmod p$;
`%challenge hash`
$c \leftarrow \mathcal{H}(A_0, B_0, A_1, B_1) \pmod q$;
`%private key`
$x \leftarrow \frac{(b_0 + c \cdot m) \cdot (1-m) - b_1 \cdot m}{a_0 \cdot (1-m) - a_1 \cdot m} \pmod q$;
`%challenges`
$c_1 \leftarrow \frac{b_1 - a_1 \cdot x}{1-m} \pmod q$;
$c_0 \leftarrow c - c_1 \pmod q$;
`%coins`
$r \leftarrow_R \mathbb{Z}_q$;
`%responses`
$f_0 \leftarrow a_0 + c_0 \cdot r \pmod q$;
$f_1 \leftarrow a_1 + c_1 \cdot r \pmod q$;
`%proof of plaintext knowledge`
$\sigma \leftarrow (A_0, B_0, c_0, f_0, A_1, B_1, c_1, f_1)$;
`%public key`
$h \leftarrow g^x \pmod p$; $pk \leftarrow (p, q, g, h)$;
`%proof of correct key construction`
$\rho \leftarrow \mathsf{ProveKey}((k, pk, \mathfrak{m}), (x, r'), k)$;
`%ciphertext`
$e \leftarrow (g^r \bmod p, h^r \cdot g^m \bmod p)$;
`%bulletin board`
$BB \leftarrow \{(e, \sigma, \sigma)\}$;
`%tally`
$\mathbf{X} \leftarrow (m, 1-m)$;
`%proof of decryption`
$\mathbf{P} \leftarrow (\mathsf{ProveDec}((pk, e, m), x, k))$;
return $((pk, \mathfrak{m}, \rho), BB, n_C, \mathbf{X}, P)$

where r' is computed such that $(pk, x, \mathfrak{m}) = \mathsf{Gen}(k; r')$.

Suppose an execution of $\mathsf{Exp\text{-}UV\text{-}Ext}(\Pi, \mathcal{A}, k)$ computes

$$\begin{aligned}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) \leftarrow \mathcal{A}(k);\\ \mathbf{Y} \leftarrow \textit{correct-tally}(pk, BB, n_C, k)\end{aligned}$$

Since $m > 1$, there is no choice $\beta \in \{1,2\}$ nor coins r such that $\mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r) \in BB$. By definition of function *correct-tally*, we have $\mathbf{Y} = (0,0)$. Moreover, since

$\mathbf{X} = (m, 1-m)$, we have $\mathbf{X} \neq \mathbf{Y}$ and $\mathbf{X}[2] = 1 - \mathbf{X}[1]$. Let us show that $\mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$. By definition of Verify, we have $PK_{\mathcal{T}}$ is a vector $(pk, \mathfrak{m}, \rho)$. Moreover, by the completeness of (ProveKey, VerKey) and (ProveDec, VerDec), we have $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) = 1$ and $\mathsf{VerDec}((pk, e, \mathbf{X}[1]), \mathbf{P}[1], k) = 1$. It remains to show that BB is the largest subset of BB satisfying the conditions given by the Tally algorithm. Since $BB = \{(e, \sigma, \sigma)\}$ and (e, σ, σ) is a vector of length $2 \cdot n_C - 1$, it suffices to show that $\mathsf{VerCiph}((pk, e, \{0, 1\}), \sigma, k) = 1$. Let us recall the definition of VerCiph (cf. [47, Figure 1] and Definition 26):

- $\mathsf{VerCiph}((pk, e, \{0, 1\}), \sigma, k)$. Parses pk as (p, q, g, h), e as (R, S), and σ as $(A_0, B_0, c_0, f_0, A_1, B_1, c_1, f_1)$, outputting 0 if parsing fails. If $g^{f_0} \equiv A_0 \cdot R^{c_0} \pmod p \wedge h^{f_0} \equiv B_0 \cdot S^{c_0} \pmod p \wedge g^{f_1} \equiv A_1 \cdot R^{c_1} \pmod p \wedge h^{f_1} \equiv B_1 \cdot (S/g)^{c_1} \pmod p \wedge \mathcal{H}(A_0, B_0, A_1, B_1) \equiv c_0 + c_1 \pmod p$, then output 1, otherwise, output 0.

We have

$$
\begin{aligned}
g^{f_0} &\equiv g^{a_0 + c_0 \cdot r} \equiv g^{a_0} \cdot (g^r)^{c_0} \equiv A_0 \cdot R^{c_0} \pmod p \\
g^{f_1} &\equiv g^{a_1 + c_1 \cdot r} \equiv g^{a_1} \cdot (g^r)^{c_1} \equiv A_1 \cdot R^{c_1} \pmod p
\end{aligned}
$$

Moreover, we have $h^{f_0} \equiv g^{x(a_0 + c_0 \cdot r)} \pmod p$ and $B_0 \cdot S^{c_0} \equiv g^{b_0 + c_0(x \cdot r + m)} \pmod p$, hence, to show $h^{f_0} \equiv B_0 \cdot S^{c_0} \pmod p$, it is sufficient to show $(b_0 + c_0 \cdot m) \equiv x \cdot a_0 \pmod q$:

$$
\begin{aligned}
& b_0 + c_0 \cdot m \\
& \equiv b_0 + c \cdot m - m \cdot c_1 \\
& \equiv b_0 + c \cdot m - \tfrac{b_1 \cdot m - a_1 \cdot m \cdot x}{1-m} \\
& \equiv \tfrac{(b_0 + c \cdot m)(1-m) - b_1 \cdot m + a_1 \cdot m \cdot x}{1-m} \\
& \equiv \frac{(b_0 + c \cdot m)(1-m) - b_1 \cdot m + \frac{a_1 \cdot m \cdot ((b_0 + c \cdot m)(1-m) - b_1 \cdot m)}{a_0(1-m) - a_1 \cdot m}}{1-m} \\
& \equiv \tfrac{(a_0(1-m) - a_1 \cdot m)((b_0 + c \cdot m)(1-m) - b_1 \cdot m)}{(1-m)(a_0(1-m) - a_1 \cdot m)} \\
& \quad + \tfrac{a_1 \cdot m((b_0 + c \cdot m)(1-m) - b_1 \cdot m)}{(1-m)(a_0(1-m) - a_1 \cdot m)} \\
& \equiv \tfrac{a_0(1-m)((b_0 + c \cdot m)(1-m) - b_1 \cdot m)}{(1-m)(a_0(1-m) - a_1 \cdot m)} \\
& \equiv \tfrac{a_0 \cdot ((b_0 + c \cdot m)(1-m) - b_1 \cdot m)}{a_0(1-m) - a_1 \cdot m} \\
& \equiv x \cdot a_0 \pmod q
\end{aligned}
$$

Similarly, $h^{f_1} \equiv g^{x(a_1 + c_1 \cdot r)} \pmod p$ and $B_1 \cdot (S/g)^{c_1} \equiv g^{b_1 + c_1(x \cdot r + m - 1)} \pmod p$, hence, to show $h^{f_1} \equiv B_1 \cdot (S/g)^{c_1} \pmod p$, it is sufficient to show $b_1 + c_1(m-1) \equiv a_1 \cdot x \pmod q$:

$$
\begin{aligned}
& b_1 + c_1(m-1) \\
& \equiv b_1 + \tfrac{(m-1)(b_1 - a_1 \cdot x)}{1-m} \\
& \equiv \tfrac{b_1(1-m) + (m-1)(b_1 - a_1 \cdot x)}{1-m} \\
& \equiv \tfrac{a_1 \cdot x(1-m)}{1-m} \\
& \equiv a_1 \cdot x \pmod q
\end{aligned}
$$

Furthermore, we have

$$
\begin{aligned}
\mathcal{H}(A_0, B_0, A_1, B_1) &\equiv c_0 + c_1 \equiv c - c_1 + c_1 \\
&\equiv \mathcal{H}(A_0, B_0, A_1, B_1) - c_1 + c_1 \pmod p
\end{aligned}
$$

It follows that $\mathsf{VerCiph}((pk, e, \{0, 1\}), \sigma, k) = 1$, concluding our proof. □

APPENDIX E
HELIOS'16 SCHEME

Generalized Helios (Figure 1) can be instantiated to derive Helios'16:

Definition 28 (Helios'16)**.** Helios'16 *is* $\mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$*, where* Γ *is additively homomorphic El Gamal [48, §2],* Σ_1 *is the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [31, Protocol 2],* Σ_2 *is the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms by Cramer et al. [47, Figure 1],* Σ_3 *is the sigma protocol for proving knowledge of equality between discrete logarithms by Chaum and Pedersen [32, §3.2], and* $\mathcal{H}$ *is a random oracle.*

Although Helios actually uses SHA-256 [101], we assume that $\mathcal{H}$ is a random oracle to prove Theorem 4. Moreover, we assume the sigma protocols used by Helios'16 satisfy the preconditions of generalized Helios—that is, [31, Protocol 2] is a sigma protocol for proving correct key construction, [47, Figure 1] is a sigma protocol for proving plaintext knowledge in a subspace, and [32, §3.2] is a sigma protocol for proving decryption. We leave formally proving this assumption as future work.

To show that Helios'16 is an election scheme, we must demonstrate that Correctness, Completeness and Injectivity are satisfied. Correctness follows immediately from Lemma 15. And we show that Completeness and Injectivity are also satisfied.

First, Completeness. Bernhard et al. [21, §4] remark that the sigma protocol used by Helios'16 to prove plaintext knowledge in a subspace satisfies satisfy special soundness and special honest verifier zero-knowledge, hence, Helios'16 satisfies Completeness by Lemma 16.

Secondly, Injectivity. A non-interactive proof system (ProveKey, VerKey) derived from a sigma protocol for proving correct key construction is sufficient to ensure that El Gamal is collision-free, assuming algorithm VerKey guarantees that public keys are constructed from suitable parameters: if $\mathsf{VerKey}((k, pk, \{0, 1\}), \rho, k) = 1$, then there exists p, q, g and h such that $pk = (p, q, g, h)$ and (p, q, g) are *cryptographic parameters*—i.e., $p = 2 \cdot q + 1$, $|q| = k$, and g is a generator of $\mathbb{Z}_p^*$ of order q.

Lemma 19. *Suppose* Σ_1 *is a sigma protocol that proves correct key construction and* $\mathcal{H}$ *is a hash function. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$*. Further suppose for all security parameters k, public keys pk, and proofs* ρ*, we have* $\mathsf{VerKey}((k, pk, \{0, 1\}), \rho, k) = 1$ *implies* $h \neq 0$ *and there exists p, q, g and h such that* $pk = (p, q, g, h)$ *and* (p, q, g) *are cryptographic parameters. It follows that additively homomorphic El Gamal is collision-free for* $\{0, 1\}$*.*

Proof. Suppose k is a security parameter, pk is a public key, ρ is a proof, $m_1, m_2 \in \{0, 1\}$ are messages and r_1 and r_2 are coins such that $\mathsf{VerKey}((k, pk, \{0, 1\}), \rho, k) = 1$, $m_1 \neq m_2 \vee r_1 \neq r_2$, $pk = (p, q, g, h)$ and (p, q, g) are cryptographic parameters, for some p, q, g and h. Further suppose that c_1 and c_2 are ciphertexts such that $c_1 = \mathsf{Enc}(pk, m_1; r_1)$, $c_2 = \mathsf{Enc}(pk, m_2; r_2)$, and Enc is El Gamal's encryption algorithm. If $r_1 \neq r_2$, then we proceed as follows. By definition of Enc, we have $c_1[1] = g^{r_1} \pmod p$ and $c_2[1] = g^{r_2} \pmod p$. Since r_1 and r_2 are distinct, we have $g^{r_1} \not\equiv g^{r_2} \pmod p$. (We implicitly assume that coins r_1 and r_2 are selected from the coin space $\mathbb{Z}_q^*$, hence, $g^{r_1} = g^{r_1} \bmod p$ and $g^{r_2} = g^{r_2} \bmod p$.) It follows that $c_1 \neq c_2$. Otherwise ($r_1 = r_2$), we have $m_1 \neq m_2$ and we proceed as follows. By definition of Enc, we have $c_1[2] = h^{r_1} \cdot g_1^m \pmod p$ and $c_2[2] = h^{r_2} \cdot g_2^m \pmod p$. Since (p, q, g) are cryptographic parameters and $h \neq 0$, we have $h^{r_1} \not\equiv h^{r_1} \cdot g \pmod p$, which is sufficient to conclude, because $m_1, m_2 \in \{0, 1\}$. □

The sigma protocol for proving knowledge of discrete logarithms by Chaum et al. [31, Protocol 2] does not explicitly require the suitability of cryptographic parameters to be checked, hence, Lemma 19 is not immediately applicable. Nonetheless, we can trivially make the necessary checks explicit and, hence, the non-interactive proof system derived from the sigma protocol for proving knowledge of discrete logarithms by Chaum et al. is sufficient to ensure that El Gamal is collision-free. It follows that Helios'16 satisfies Injectivity, hence, Helios'16 is an election scheme.

APPENDIX F
PROOF: HELIOS'16 IS VERIFIABLE

Elections schemes constructed from generalized Helios satisfy individual (§F-A) and universal (§F-B) verifiability, hence, such schemes satisfy election verifiability with external authentication (§F-C). It follows that Helios'16 satisfies election verifiability (§F-D).

A. Individual verifiability

Proposition 20. *Suppose Γ, Σ_1, Σ_2, Σ_3 and $\mathcal{H}$ satisfy the preconditions of Figure 1. Further suppose that Γ is collision-free for $\{0, 1\}$. We have $\mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ satisfies individual verifiability.*

The proof of Proposition 20 is similar to the proof of Lemma 17.

Proof. Let $\mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ and $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$. Suppose k is a security parameter, $PK_{\mathcal{T}}$ is a public key, n_C is an integer, and β and β' are choices. Further suppose that b and b' are ballots such that $b \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k)$, $b' \leftarrow \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta', k)$, $b \neq \perp$, and $b' \neq \perp$. By definition of Vote, we have $PK_{\mathcal{T}}$ parses as a vector $(pk, \mathfrak{m}, \rho)$ and $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) = 1$. Moreover, $b[1]$ and $b'[1]$ are ciphertexts such that $b[1] \leftarrow \mathsf{Enc}(pk, m)$ and $b'[1] \leftarrow \mathsf{Enc}(pk, m')$, where $m, m' \in \{0, 1\}$. Furthermore, the ciphertexts are constructed using random coins—i.e., the coins used by $b[1]$ and $b'[1]$ will be distinct with overwhelming probability. Since Γ is collision-free for $\{0, 1\}$, we have $b[1] \neq b'[1]$ and $b \neq b'$ with overwhelming probability, concluding our proof. □

B. Universal verifiability

Proposition 21. *Suppose Γ, Σ_1, Σ_2, Σ_3 and $\mathcal{H}$ satisfy the preconditions of Figure 1. Further suppose that Σ_1, Σ_2 and Σ_3 satisfy special soundness and special honest verifier zero-knowledge, and $\mathcal{H}$ is a random oracle. We have $\mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ satisfies universal verifiability.*

Proof. Let $\Pi = \mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$, $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, and $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$. By Theorem 12, each of the non-interactive proof systems satisfies simulation sound extractability.

Suppose k is a security parameter and $\mathcal{A}$ is a PPT adversary. Further suppose that an execution of $\mathsf{Exp\text{-}UV\text{-}Ext}(\Pi, \mathcal{A}, k)$ computes

$$(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) \leftarrow \mathcal{A}(k);$$
$$\mathbf{Y} \leftarrow \mathit{correct\text{-}tally}(PK_{\mathcal{T}}, BB, n_C, k)$$

such that $\mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$. (If $\mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) \neq 1$, then we can conclude immediately.) We focus on the case $n_C > 1$; the case $n_C = 1$ is similar.

By definition of the verification algorithm, vector $\mathbf{X}$ is of length n_C and P is a vector of length $n_C - 1$. Moreover, $PK_{\mathcal{T}}$ is a vector $(pk, \mathfrak{m}, \rho)$. Let $\{b_1, \ldots, b_\ell\}$ be the largest subset of BB such that for all $1 \leq i \leq \ell$ we have b_i is a vector of length $2 \cdot n_C - 1$ and $\bigwedge_{j=1}^{n_C - 1} \mathsf{VerCiph}((pk, b_i[j], \{0, 1\}), b_i[j + n_C - 1], j, k) = 1 \wedge \mathsf{VerCiph}((pk, b_i[1] \otimes \cdots \otimes b_i[n_C - 1], \{0, 1\}), b_i[2 \cdot n_C - 1], n_C, k) = 1$.

We have for all choices $\beta \in \{1, \ldots, n_C\}$, coins r and ballots $b = \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r)$ that $b \notin BB \setminus \{b_1, \ldots, b_\ell\}$ with overwhelming probability, since such an occurrence would imply a contradiction: $\{b_1, \ldots, b_\ell\}$ is not the largest subset of BB satisfying the conditions given by the tally algorithm, because b is a vector of length $2 \cdot n_C - 1$ such that $\bigwedge_{j=1}^{n_C - 1} \mathsf{VerCiph}((pk, b[j], \{0, 1\}), b[j + n_C - 1], j, k) = 1 \wedge \mathsf{VerCiph}((pk, b[1] \otimes \cdots \otimes b[n_C - 1], \{0, 1\}), b[2 \cdot n_C - 1], n_C, k) = 1$ with overwhelming probability, but $b \notin \{b_1, \ldots, b_\ell\}$. It follows that:

$$\begin{aligned} &\mathit{correct\text{-}tally}(PK_{\mathcal{T}}, BB, n_C, k) \\ &\qquad = \mathit{correct\text{-}tally}(PK_{\mathcal{T}}, \{b_1, \ldots, b_\ell\}, n_C, k) \end{aligned} \quad (1)$$

A proof of (1) follows from the definition of function *correct-tally*.

We proceed by distinguishing two cases.

Case I: $\{b_1, \ldots, b_\ell\} = \emptyset$. By definition of function *correct-tally* and (1), we have $\mathbf{Y}$ is a vector of length n_C such that $\bigwedge_{j=1}^{n_C} \mathbf{Y}[j] = 0$. Since $\bigwedge_{i=j}^{n_C} \mathbf{X}[j] = 0$, we have $\mathbf{X} = \mathbf{Y}$ by definition of the verification algorithm.

Case II: $\{b_1, \ldots, b_\ell\} \neq \emptyset$. By definition of the verification algorithm, we have $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) = 1$. Moreover, by simulation sound extractability, we are assured that pk is an output of Gen with overwhelming probability—i.e., there exists s and sk such that $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(k; s)$.

By simulation sound extractability, with overwhelming probability, for all $1 \leq i \leq \ell$ there exists messages $m_{i,1}, \ldots, m_{i,n_C-1} \in \{0,1\}$ and coins $r_{i,1}, \ldots, r_{i,2\cdot n_C-2}$ such that for all $1 \leq j \leq n_C - 1$ we have

$$b_i[j+n_C-1] = \mathsf{ProveCiph}((pk, b_i[j], \{0,1\}), (m_{i,j}, r_{i,j}), j, k; r_{i,j+n_C-1})$$

and

$$b_i[j] = \mathsf{Enc}(pk, m_{i,j}; r_{i,j}).$$

Moreover, for all $1 \leq i \leq \ell$ we have $\sum_{j=1}^{n_C-1} m_{i,j} \in \{0,1\}$ and there exist coins $r_{i,2\cdot n_C-1}$ such that

$$b_i[2 \cdot n_C - 1] = \mathsf{ProveCiph}(pk, c, \{0,1\}), (m, r), n_C, k; r_{i,2\cdot n_C-1})$$

with overwhelming probability, where $c \leftarrow b_i[1] \otimes \cdots \otimes b_i[n_C - 1]$, $m \leftarrow m_{i,1} \odot \cdots \odot m_{i,n_C-1}$, and $r \leftarrow r_{i,1} \oplus \cdots \oplus r_{i,n_C-1}$.

By inspection of Vote, for all $1 \leq i \leq \ell$ there exists β_i, r_i such that

$$b_i = \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta_i, k; r_i)$$

and either $\beta_i = n_C \wedge \bigwedge_{j=1}^{n_C-1} m_{i,j} = 0$ or $\beta_i \in \{1, \ldots, n_C - 1\} \wedge m_{i,\beta_i} = 1 \wedge \bigwedge_{j \in \{1,\ldots,\beta_i-1,\beta_i+1,\ldots,n_C-1\}} m_{i,j} = 0$. It follows for all $1 \leq i \leq \ell$ and $1 \leq j \leq n_C - 1$ that:

$$m_{i,j} = 0 \Longleftrightarrow \beta_i = n_C \vee \beta_i \neq j \tag{2}$$

$$m_{i,j} = 1 \Longleftrightarrow \beta_i = j \tag{3}$$

Moreover, for all $1 \leq i \leq \ell$ we have:

$$\sum_{j=1}^{n_C-1} m_{i,j} = 0 \Longleftrightarrow \beta_i = n_C \tag{4}$$

Furthermore, we have the following facts:

Fact 1. *For all integers β and k such that $1 \leq \beta \leq n_C$, we have:*

$$\exists^{=k} b \in (\{b_1, \ldots, b_\ell\} \setminus \{\perp\}) : \exists r : b = \mathsf{Vote}(PK_{\mathcal{T}}, n_C, \beta, k; r) \iff \exists^{=k} i \in \{1, \ldots, \ell\} : \beta = \beta_i$$

Fact 2. *For all integers j and k such that $1 \leq j \leq n_C - 1$, we have:*

$$\exists^{=k} i \in \{1, \ldots, \ell\} : \beta_i = j \iff k = \sum_{i=1}^{\ell} m_{i,j}$$

Proof of Fact 2. For the forward implication, suppose j, k are integers such that $1 \leq j \leq n_C - 1$ and $\exists^{=k} i \in \{1, \ldots, \ell\} : \beta_i = j$. We proceed by induction on ℓ. In the base case ($\ell = 0$), we have $k = 0$, hence, $k = \sum_{i=1}^{\ell} m_{i,j}$. In the inductive case, we distinguish two cases. Case I: $\exists^{=k} i \in \{1, \ldots, \ell - 1\} : \beta_i = j$ holds. We have $\beta_\ell \neq j$ by definition of the counting quantifier and, hence, $m_{i,j} = 0$ by (2). By our induction hypothesis, we derive $k = \sum_{i=1}^{\ell-1} m_{i,j} = \sum_{i=1}^{\ell} m_{i,j}$. Case II: $\exists^{=k} i \in \{1, \ldots, \ell - 1\} : \beta_i = j$ does not hold. We have $\beta_\ell = j$ by definition of the counting quantifier and, hence, $m_{i,j} = 1$ by (3). Moreover, we have $\exists^{=k-1} i \in \{1, \ldots, \ell - 1\} : \beta_i = j$ holds. By our induction hypothesis, we derive $k - 1 = \sum_{i=1}^{\ell-1} m_{i,j}$, that is, $k = \sum_{i=1}^{\ell} m_{i,j}$.

For the reverse implication, suppose j, k are integers such that $1 \leq j \leq n_C - 1$ and $k = \sum_{i=1}^{\ell} m_{i,j}$. We proceed by induction on ℓ. In the base case ($\ell = 0$), we have $k = 0$, hence, $\exists^{=k} i \in \{1, \ldots, \ell\} : \beta_i = j$. In the inductive case, we distinguish two cases. Case I: $k = \sum_{i=1}^{\ell-1} m_{i,j}$. We have $m_{\ell,j} = 0$, hence, $\beta_\ell \neq j$ by (2). By our induction hypothesis, we have $\exists^{=k} i \in \{1, \ldots, \ell - 1\} : \beta_i = j$. Since $\beta_\ell \neq j$, the result follows. Case II: $k \neq \sum_{i=1}^{\ell-1} m_{i,j}$. Since $m_{\ell,j} \in \{0,1\}$, we have $m_{\ell,j} = 1$, hence, $\beta_\ell = j$ by (3). Moreover, we have $k - 1 = \sum_{i=1}^{\ell-1} m_{i,j}$. By our induction hypothesis, we derive $\exists^{=k-1} i \in \{1, \ldots, \ell - 1\} : \beta_i = j$. The result follows.

Fact 3. *For all integers k, we have*

$$\exists^{=k} i \in \{1, \ldots, \ell\} : \beta_i = n_C \iff k = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$$

Proof of Fact 3. For the forward implication, suppose $\exists^{=k} i \in \{1, \ldots, \ell\} : \beta_i = n_C$. We proceed by induction on ℓ. In the base case ($\ell = 0$), we have $k = 0$, hence, $k = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$. In the inductive case, we distinguish two cases. Case I: $\exists^{=k} i \in \{1, \ldots, \ell - 1\} : \beta_i = n_C$ holds. We have $\beta_\ell \neq n_C$ by definition of the counting quantifier and we derive $\sum_{j=1}^{n_C-1} m_{\ell,j} \neq 0$ by (4). Moreover, since $\sum_{j=1}^{n_C-1} m_{\ell,j} \in \{0,1\}$, we have $\sum_{j=1}^{n_C-1} m_{\ell,j} = 1$. By our induction hypothesis, we derive $k = \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j} = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$. Case II: $\exists^{=k} i \in \{1, \ldots, \ell - 1\} : \beta_i = n_C$ does not hold. We have $\beta_\ell = n_C$ by definition of the counting quantifier and we derive $\sum_{j=1}^{n_C-1} m_{i,j} = 0$ by (4). Moreover, we have $\exists^{=k-1} i \in \{1, \ldots, \ell - 1\} : \beta_i = n_C$ holds. By our induction hypothesis, we derive $k - 1 = \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j}$, that is, $k = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j} = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$.

For the reverse implication, suppose $k = \ell - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell} m_{i,j}$. We proceed by induction on ℓ. In the base case ($\ell = 0$), we have $k = 0$, hence, $\exists^{=k} i \in \{1, \ldots, \ell\} : \beta_i = n_C$. In the inductive case, we distinguish two cases. Case I: $k = \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j}$. We have $\sum_{j=1}^{n_C-1} m_{\ell,j} = 1$. Since $m_{\ell,1}, \ldots, m_{\ell,n_C-1} \in \{0,1\}$, there exists j such that $1 \leq j \leq n_C - 1$ and $m_{\ell,j} = 1$, moreover, $\beta_\ell = j$ by (3), hence, $\beta_\ell \neq n_C$. By our induction hypothesis, we derive $\exists^{=k} i \in \{1, \ldots, \ell - 1\} : \beta_i = n_C$. The result follows. Case II: $k \neq \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j}$. Since $\sum_{j=1}^{n_C-1} m_{\ell,j} \in \{0,1\}$, we have $\sum_{j=1}^{n_C-1} m_{\ell,j} = 0$, and we derive $\beta_i = n_C$ by (4). Moreover, we have $k - 1 = \ell - 1 - \sum_{j=1}^{n_C-1} \sum_{i=1}^{\ell-1} m_{i,j}$. By our induction hypothesis, we derive $\exists^{=k-1} i \in \{1, \ldots, \ell - l\} : \beta_i = n_C$. The result follows.

We proceed the proof of Proposition 21 using the above facts.

By definition of the verification algorithm, we have $\bigwedge_{j=1}^{n_C-1} \mathsf{VerDec}((pk, b_1[j] \otimes \cdots \otimes b_\ell[j], \mathbf{X}[j]), P[j], k) = 1 \wedge \mathbf{X}[n_C] = \ell - \sum_{j=1}^{n_C-1} \mathbf{X}[j]$. By simulation sound extractability, we have for all $1 \leq j \leq n_C - 1$ that $\mathbf{X}[j] = \mathsf{Dec}(sk, b_1[j] \otimes \cdots \otimes b_\ell[j])$ with overwhelming probability, hence, $\mathbf{X}[j] = m_{1,j} \odot \cdots \odot m_{\ell,j}$, with overwhelming probability. Let m_B be the largest integer such that $\{0, \ldots, m_B\} \subseteq \mathfrak{m}$. By definition of the verification algorithm, we have $\ell \leq m_B$. It follows that $m_{1,j} \odot \cdots \odot m_{\ell,j} = \sum_{i=1}^{\ell} m_{i,j}$, hence,

$$\mathbf{X}[j] = \sum_{i=1}^{\ell} m_{i,j}$$

with overwhelming probability. By definition of function *correct-tally*, (1) and Fact 1, we have $\mathbf{Y}$ is a vector of length n_C such that for all $1 \leq \beta \leq n_C$ we have

$$\mathbf{Y}[\beta] = k \text{ if } \exists^{=k} i \in \{1, \ldots, \ell\} : \beta = \beta_i$$

It follows by Facts 2 and 3 that for all $1 \leq \beta \leq n_C$ we have $\mathbf{X}[\beta] = \mathbf{Y}[\beta]$ with overwhelming probability, hence, $\mathbf{X} = \mathbf{Y}$ with overwhelming probability.

We have $\mathbf{X} = \mathbf{Y}$ with overwhelming probability in both cases—i.e., $\mathsf{Exp\text{-}UV\text{-}Ext}(\Pi, \mathcal{A}, k)$ outputs 0 with overwhelming probability and $\mathsf{Succ}(\mathsf{Exp\text{-}UV\text{-}Ext}(\Pi, \mathcal{A}, k))$ is negligible, concluding our proof. □

C. Election verifiability

By Propositions 20 & 21, election schemes constructed from generalized Helios satisfy election verifiability with external authentication:

Corollary 22. *Suppose Γ, Σ_1, Σ_2, Σ_3 and $\mathcal{H}$ satisfy the preconditions of Figure 1. Further suppose that Γ is collision-free for $\{0,1\}$, Σ_1, Σ_2 and Σ_3 satisfy special soundness and special honest verifier zero-knowledge, and $\mathcal{H}$ is a random oracle. We have $\mathsf{Helios}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \mathcal{H})$ satisfies election verifiability with external authentication.*

D. Proof: Theorem 4

Our proof of Theorem 4 is reliant on Corollary 22. We have already shown that the sigma protocol used by Helios'16 to prove discrete logarithms is sufficient to ensure that El Gamal is collision-free (Lemma 19), hence, it remains to show that the sigma protocols used by Helios'16 satisfy special soundness and special honest verifier zero-knowledge.

Bernhard et al. [21, §4] remark that the sigma protocols used by Helios'16 to prove discrete logarithms and equality between discrete logarithms both satisfy special soundness and special honest verifier zero-knowledge, hence, Theorem 12 is applicable. Bernhard et al. also remark that the sigma protocol for proving knowledge of disjunctive equality between discrete logarithms satisfies special soundness and "almost special honest verifier zero-knowledge" and argue that "we could fix this[, but] it is easy to see that ... all relevant theorems [including Theorem 12] still hold." We adopt the same and assume that Theorem 12 is applicable.

Proof of Theorem 4. The proof follows from Corollary 22, subject to the applicability of Theorem 12 to the sigma protocol used by Helios'16 to prove knowledge of disjunctive equality between discrete logarithms. □

APPENDIX G
PROOF: Exp-EV-Int ⇒ Exp-IV-Int

Our eligibility verifiability experiment (§IV-B3) asserts that no one can construct a ballot that appears to be associated with public credential pk unless they know private credential sk. It follows that a voter can uniquely identify her ballot on the bulletin board, because no one else knows her private credential. Eligibility verifiability therefore implies individual verifiability (Theorem 6).

Our proof of Theorem 6 is reliant on distinct credentials, which is an consequence of eligibility verifiability:

Lemma 23. *If an election scheme Π satisfies strong eligibility verifiability, then there exists a negligible function μ, such that for all security parameters k, we have*

$$\begin{aligned} Pr[(PK_\mathcal{T}, SK_\mathcal{T}, m_B, m_C) \leftarrow \mathsf{Setup}(k);& \\ (pk_0, sk_0) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k);& \\ (pk_1, sk_1) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k) :& \\ sk_0 = sk_1] \leq \mu(k)& \end{aligned}$$

Proof. Suppose an election scheme Π satisfies Exp-EV-Int, but

$$\begin{aligned} Pr[(PK_\mathcal{T}, SK_\mathcal{T}, m_B, m_C) \leftarrow \mathsf{Setup}(k);& \\ (pk_0, sk_0) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k);& \\ (pk_1, sk_1) \leftarrow \mathsf{Register}(PK_\mathcal{T}, k) :& \\ sk_0 = sk_1] \geq \frac{1}{p(k)}& \end{aligned}$$

for some polynomial function p and security parameter k. Then we can construct an adversary $\mathcal{A}$ that wins Exp-EV-Int as follows. Adversary $\mathcal{A}$ is given input k and runs Setup to obtain a key pair $(PK_\mathcal{T}, SK_\mathcal{T})$, chooses some positive integer n_V, and outputs $(PK_\mathcal{T}, n_V)$. The challenger then generates n_V key pairs and gives the set L of public keys to $\mathcal{A}$. Now $\mathcal{A}$ simply runs $\mathsf{Register}(PK_\mathcal{T}, k)$ to get a key pair (pk, sk), chooses some positive integers n_C and β such that $1 \leq \beta \leq n_C$, computes $b \leftarrow \mathsf{Vote}(sk, PK_\mathcal{T}, n_C, \beta, k)$, and outputs (n_C, b). We know that secret keys generated by Register collide with probability at least $\frac{1}{p(k)}$, so Register must generate a particular secret key sk' with probability $\frac{1}{p(k)}$. Therefore, this sk' will correspond to one of the public keys in L with probability $\frac{n_V}{p(k)}$. Furthermore, the key sk generated by the adversary will be sk' with probability $\frac{1}{p(k)}$. Therefore, b will be a vote constructed under a voter's secret key with probability $\frac{n_V}{p(k)^2}$, so $\mathcal{A}$ wins the experiment with non-negligible probability. □

A. Proof: Theorem 6

Suppose there exists an adversary $\mathcal{A}'$ that wins $\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}', k)$ with probability $\frac{1}{p(k)}$ for some polynomial function p. Then we can construct an adversary $\mathcal{A}$ that wins $\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, k)$ with non-negligible probability. Adversary $\mathcal{A}$ is given k as input, which it passes to $\mathcal{A}'$. Adversary $\mathcal{A}'$ may ask for secret keys from its oracle C, in which case $\mathcal{A}$ forwards these queries to its own, identical oracle. Adversary $\mathcal{A}$ then forwards the oracle's response back to $\mathcal{A}'$. Adversary $\mathcal{A}'$ then outputs $(PK_{\mathcal{T}}, n_V)$, which is then output by $\mathcal{A}$. Next, $\mathcal{A}$ is given the public keys $(pk_1, \ldots, pk_{n_V})$. Adversary $\mathcal{A}$ passes these keys to $\mathcal{A}'$, which returns $(n_C, \beta, \beta', i, j)$. Any oracle queries made by $\mathcal{A}'$ are handled exactly as before. Now $\mathcal{A}$ queries its oracle C on i. The oracle returns sk_i. Adversary $\mathcal{A}$ computes $b = \mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta)$ and outputs (n_C, β', j, b). Adversary $\mathcal{A}'$ wins $\mathsf{Exp\text{-}IV\text{-}Int}(\Pi, \mathcal{A}, k)$ with non-negligible probability, so with non-negligible probability $b = \mathsf{Vote}(sk_j, PK_{\mathcal{T}}, n_C, \beta')$ and $\mathcal{A}'$ (and therefore $\mathcal{A}$) did not query the oracle on input j. Adversary $\mathcal{A}$ only makes one additional oracle query on input i, so again, $\mathcal{A}$ does not query the oracle on j. Furthermore, by Lemma 23, $sk_i = sk_j$ with only negligible probability. Therefore $\mathcal{A}$ wins $\mathsf{Exp\text{-}EV\text{-}Int}(\Pi, \mathcal{A}, k)$ with probability $\frac{1}{p(k)} - \mathsf{negl}(k)$. □

Appendix H
JCJ Scheme

We formalize a generic construction for JCJ-like election schemes (Figure 3). Our construction is parameterized on the choice of homomorphic encryption scheme and sigma protocols.[46] The specification of algorithms Setup, Register and Vote follow from our informal descriptions (§VI).[47] The tallying algorithm performs the following steps:

1) *Remove invalid ballots:* The tallier discards any ballots from the bulletin board for which proofs do not hold.
2) *Eliminating duplicates:* The tallier performs pairwise PETs on the encrypted credentials and discard any ballots for which a test holds, that is, ballots using the same credential are discarded.[48]
3) *Mixing:* The tallier mixes the ciphertexts in the ballots (i.e., the encrypted choices and the encrypted credentials), using the same secret permutation for both mixes, hence, the mix preserves the relation between encrypted choices and credentials. Let $\mathbf{C_1}$ and $\mathbf{C_2}$ be the vectors output by these mixes. The tallier also mixes the public credentials published by the registrar. Let $\mathbf{C_3}$ be the vector output by this mix.
4) *Remove ineligible ballots:* The tallier discards ciphertexts $\mathbf{C_1}[i]$ from $\mathbf{C_1}$ if there is no ciphertext c in $\mathbf{C_3}$ such that a PET holds for c and $\mathbf{C_2}[i]$, that is, ballots cast using ineligible credentials are discarded.
5) *Decrypting:* The tallier decrypts the remaining encrypted choices in $\mathbf{C_1}$ and proves that decryption was performed correctly. The tallier identifies the winning candidate from the decrypted choices.

The Verify algorithm checks that each of the above steps has been performed correctly.

Lemmata 24–26 demonstrate that generalized JCJ is a construction for election schemes.

Lemma 24. *Suppose* Γ, Σ_1, Σ_2, Σ_3, $\Sigma 4$, Σ_5, Σ_6 *and* $\mathcal{H}$ *satisfy the preconditions of Figure 3. We have* $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$ *satisfies Correctness.*

Proof. Our proof is by induction on the number of ballots n_B. We start with the base case, $n_B = 1$. For all k, n_C, and $\beta \in \{1, \ldots, n_C\}$, we have

$$\begin{aligned}
&(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k);\\
&(pk, sk) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k);\\
&b \leftarrow \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k);\\
&\mathbf{Y}[\beta] \leftarrow \mathbf{Y}[\beta] + \mathbf{1};\\
&L \leftarrow \{pk\};\\
&BB \leftarrow \{b\};\\
&(\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, L, n_C, k);
\end{aligned}$$

Assume $n_C \le m_C$ (otherwise, we trivially satisfy correctness). Hence, we need to show $\mathbf{X}[\beta] = 1$ and $\mathbf{X}[i] = 0$ for all $i \neq \beta$. By definition of Setup, we have $PK_{\mathcal{T}} = (pk_T, \mathfrak{m}, \rho)$, $SK_{\mathcal{T}} = (pk_T, sk_T)$ and $m_C = |\mathfrak{m}|$. By definition of Vote, we have $b = (c_1, c_2, \sigma, \tau)$, where $c_1 = \mathsf{Enc}(pk_T, \beta; r_1)$, $c_2 = \mathsf{Enc}(pk_T, sk; r_2)$, $\sigma = \mathsf{ProveCiph}((pk_T, c_1, \{1, \ldots, n_C\}), (\beta, r_1), k)$, and $\tau = \mathsf{ProveBind}((pk_T, c_1, c_2,), (\beta, r_1, sk, r_2), k)$. Since $\beta \in \{1, \ldots, n_C\}$ and $n_C \le |\mathfrak{m}|$, we have β is a message in Γ's message space

- *Remove invalid ballots:* This involves checking the proofs σ and τ. Since they were honestly computed, they verify with overwhelming probability.
- *Remove duplicate ballots:* Tally would check here if there are multiple ballots computed using the same secret key. Since there is only one ballot, this check passes trivially.
- *Mixing:* Tally mixes the ballots. Since there is only one ballot, Tally will just re-encrypt the ballot. Let the re-encryptions of $b[1]$ and $b[2]$ be $b'[1]$ and $b'[2]$, respectively. This is done honestly, so $b'[1]$ will still be an encryption of β and $b'[2]$ will still be an encryption of sk.
- *Remove ineligible ballots:* As mentioned, $b'[2]$ is still an encryption of sk, which is a valid secret key, so the ballot is not eliminated.
- *Decrypting:* Finally, Tally computes $\beta' \leftarrow \mathsf{Dec}(sk_T, b'[1])$. Again, since $b'[1]$ is still an encryption of β, we have $\beta' = \beta$. Tally then increments $\mathbf{X}[\beta]$ by 1.

Since we now have $\mathbf{X}[\beta] = 1$ and $\mathbf{X}[i] = 0$ for all $i \neq \beta$, we have that JCJ satisfies correctness when $n_B = 1$.

Now we assume that JCJ is correct for $n_B = n$, and prove that it satisfies correctness for $n_B = n + 1$. First, we note that since we are only adding one more vote, and therefore only registering one more key pair, the probability that $sk_{n+1} = sk_i$ for some $i \in \{1, \ldots, n_B\}$ is negligible,

46. For brevity, the encryption scheme's message space $\mathfrak{m}$ is assumed to contain $\{1, \ldots, |\mathfrak{m}|\}$.

47. Algorithm Setup bounds the maximum number of voters to a polynomial in the security parameter to ensure that private voter credentials do not collide, with overwhelming probability.

48. JCJ defines discarding ballots in accordance with a revoting policy [83, §4.1]. However, we have shown that JCJ fails to satisfy universal verifiability when the policy proposed by Juels et al. is adopted (§IV-B2). So, we consider a policy that discards ballots using the same credential—i.e., choices by voters that cast multiple ballots will be discarded.

Fig. 3 Generalized JCJ

Suppose $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a multiplicatively homomorphic asymmetric encryption scheme with a message space over $\mathbb{Z}_m^*$ for some integer m determined by the security parameter, $\mathfrak{e}$ is an identity element of Γ's message space with respect to $\odot$, Σ_1 proves correct key construction, Σ_2 proves plaintext knowledge in a subspace, Σ_3 proves conjunctive plaintext knowledge, Σ_4 proves correct decryption, Σ_5 is a PET, Σ_6 is a mixnet, and $\mathcal{H}$ is a hash function. Let $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_2, \mathcal{H}) = (\mathsf{ProveCiph}, \mathsf{VerCiph})$, $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveBind}, \mathsf{VerBind})$, $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$, $\mathsf{FS}(\Sigma_5, \mathcal{H}) = (\mathsf{ProvePET}, \mathsf{VerPET})$, and $\mathsf{FS}(\Sigma_6, \mathcal{H}) = (\mathsf{ProveMix}, \mathsf{VerMix})$. We define *generalized JCJ* $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ as follows.

- $\mathsf{Setup}(k)$. Select coins r, compute $(pk_T, sk_T, \mathfrak{m}) \leftarrow \mathsf{Gen}(k; r); \rho \leftarrow \mathsf{ProveKey}((k, pk_T, \mathfrak{m}), (sk_T, r), k); PK_{\mathcal{T}} \leftarrow (pk_T, \mathfrak{m}, \rho); SK_{\mathcal{T}} \leftarrow (pk_T, sk_T); m_C \leftarrow |\mathfrak{m}|$, and output $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, poly(k), m_C)$.
- $\mathsf{Register}(PK_{\mathcal{T}}, k)$. Parse $PK_{\mathcal{T}}$ as $(pk_T, \mathfrak{m}, \rho)$, outputting $(\perp, \perp)$ if parsing fails or $\mathsf{VerKey}((k, pk_T, \mathfrak{m}), \rho, k) \neq 1$. Compute $d \leftarrow_R \mathfrak{m}; pd \leftarrow \mathsf{Enc}(pk_T, d)$ and output (d, pd).
- $\mathsf{Vote}(d, PK_{\mathcal{T}}, n_C, \beta, k)$. Parse $PK_{\mathcal{T}}$ as a vector $(pk_T, \mathfrak{m}, \rho)$, outputting $\perp$ if parsing fails or $\mathsf{VerKey}((k, pk_T, \mathfrak{m}), \rho, k) \neq 1 \vee \beta \notin \{1, \dots, n_C\} \vee \{1, \dots, n_C\} \not\subseteq \mathfrak{m}$. Select coins r_1 and r_2, compute $c_1 \leftarrow \mathsf{Enc}(pk_T, \beta; r_1); c_2 \leftarrow \mathsf{Enc}(pk_T, d; r_2); \sigma \leftarrow \mathsf{ProveCiph}((pk_T, c_1, \{1, \dots, n_C\}), (\beta, r_1), k); \tau \leftarrow \mathsf{ProveBind}((pk_T, c_1, c_2), (\beta, r_1, d, r_2), k)$ and output ballot (c_1, c_2, σ, τ).
- $\mathsf{Tally}(SK_{\mathcal{T}}, BB, L, n_C, k)$. Initialize vectors $\mathbf{X}$ of length n_C and $\mathbf{P}$ of length 9. Parse $SK_{\mathcal{T}}$ as (pk_T, sk_T). Compute **for** $1 \le j \le n_C$ **do** $\mathbf{X}[j] \leftarrow 0$. Proceed as follows.
  1) *Remove invalid ballots:* Let $\{b_1, \dots, b_\ell\}$ be the largest subset of BB such that for all $1 \le i \le \ell$ we have b_i is a vector of length 4 and $\mathsf{VerCiph}((pk_T, b_i[1], \{1, \dots, n_C\}), b_i[3], k) = 1 \wedge \mathsf{VerBind}((pk_T, b_i[1], b_i[2]), b_i[4], k) = 1$. If $\{b_1, \dots, b_\ell\} = \emptyset$, then output $(\mathbf{X}, \mathbf{P})$.
  2) *Eliminating duplicates:* Initialize $\mathbf{P_{dupl}}$ as a vector of length ℓ. For each $1 \le i \le \ell$, if there exists σ and $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$ such that $\sigma \leftarrow \mathsf{ProvePET}((pk_T, b_i[2], b_j[2], 1), sk_T, k)$ and $\mathsf{VerPET}((pk_T, b_i[2], b_j[2], 1), \sigma, k) = 1$, then assign $\mathbf{P_{dupl}}[i] \leftarrow (\sigma)$; otherwise, compute $\sigma_j \leftarrow \mathsf{ProvePET}((pk_T, b_i[2], b_j[2], 0), sk_T, k)$ for each $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$ and assign $\mathbf{P_{dupl}}[i] \leftarrow (\sigma_1, \dots, \sigma_{i-1}, \sigma_{i+1}, \dots, \sigma_\ell)$. Let $\mathbf{BB}$ be the empty vector and compute **for** $1 \le i \le \ell \wedge |\mathbf{P_{dupl}}[i]| = \ell - 1$ **do** $\mathbf{BB} \leftarrow \mathbf{BB} \parallel (b_i)$, where $\mathbf{BB} \parallel (b_i)$ denotes the concatenation of vectors $\mathbf{BB}$ and (b_i)—i.e., $\mathbf{BB} \parallel (b_i) = (\mathbf{BB}[1], \dots, \mathbf{BB}[|\mathbf{BB}|], b_i)$.
  3) *Mixing:* Suppose $\mathbf{BB} = (b'_1, \dots, b'_{|\mathbf{BB}|})$. Select a random permutation χ on $\{1, \dots, |\mathbf{BB}|\}$, initialize $\mathbf{C_1}$, $\mathbf{C_2}$, $\mathbf{r_1}$ and $\mathbf{r_2}$ as vectors of length $|\mathbf{BB}|$, and fill $\mathbf{r_1}$ and $\mathbf{r_2}$ with random coins. Compute **for** $1 \le i \le |\mathbf{BB}|$ **do** $\mathbf{C_1}[\chi(i)] \leftarrow b'_i[1] \otimes \mathsf{Enc}(PK_{\mathcal{T}}, \mathfrak{e}; \mathbf{r_1}[i])$; $\mathbf{C_2}[\chi(i)] \leftarrow b'_i[2] \otimes \mathsf{Enc}(PK_{\mathcal{T}}, \mathfrak{e}; \mathbf{r_2}[i])$ and $P_{mix,1} \leftarrow \mathsf{ProveMix}((pk_T, (b'_1[1], \dots, b'_{|\mathbf{BB}|}[1]), \mathbf{C_1}), (\mathbf{r_1}, \chi), k); P_{mix,2} \leftarrow \mathsf{ProveMix}((pk_T, (b'_1[2], \dots, b'_{|\mathbf{BB}|}[2]), \mathbf{C_2}), (\mathbf{r_2}, \chi), k)$. Similarly, select a random permutation χ' on $\{1, \dots, |L|\}$, initialize $\mathbf{C_3}$ and $\mathbf{r_3}$ as vectors of length $|L|$, fill $\mathbf{r_3}$ with random coins, and compute **for** $1 \le i \le |L|$ **do** $\mathbf{C_3}[\chi'(i)] \leftarrow L[i] \otimes \mathsf{Enc}(PK_{\mathcal{T}}, \mathfrak{e}; \mathbf{r_3}[i])$ and $P_{mix,3} \leftarrow \mathsf{ProveMix}((pk_T, L), (\mathbf{r_3}, \chi'), k)$.
  4) *Remove ineligible ballots:* Initialize $\mathbf{P_{inelig}}$ as a vector of length $|\mathbf{C_2}|$. For each $1 \le i \le |\mathbf{C_2}|$, if there exists σ and $c \in \mathbf{C_3}$ such that $\sigma \leftarrow \mathsf{ProvePET}((pk_T, \mathbf{C_2}[i], c, 1), sk_T, k)$ and $\mathsf{VerPET}((pk_T, \mathbf{C_2}[i], c), \sigma, k) = 1$, then assign $\mathbf{P_{inelig}}[i] \leftarrow (\sigma)$; otherwise, compute $\sigma_j \leftarrow \mathsf{ProvePET}((pk_T, \mathbf{C_2}[i], \mathbf{C_3}[j], 0), sk_T, k)$ for each $j \in \{1, \dots, |\mathbf{C_3}|\}$ and assign $\mathbf{P_{inelig}}[i] \leftarrow (\sigma_1, \dots, \sigma_{|\mathbf{C_3}|})$.
  5) *Decrypting:* Initialize $\mathbf{P_{dec}}$ as the empty vector. Compute **for** $1 \le i \le |\mathbf{C_1}| \wedge |\mathbf{P_{inelig}}[i]| = 1$ **do** $\beta \leftarrow \mathsf{Dec}(sk_T, \mathbf{C_1}[i])$; $\sigma \leftarrow \mathsf{ProveDec}(pk_T, \mathbf{C_1}[i], \beta), sk_T, k)$; $\mathbf{X}[\beta] \leftarrow \mathbf{X}[\beta] + 1$; $\mathbf{P_{dec}} \leftarrow \mathbf{P_{dec}} \parallel (\sigma)$.

  Assign $\mathbf{P} \leftarrow (\mathbf{P_{dupl}}, \mathbf{C_1}, P_{mix,1}, \mathbf{C_2}, P_{mix,2}, \mathbf{C_3}, P_{mix,3}, \mathbf{P_{inelig}}, \mathbf{P_{dec}})$ and output $(\mathbf{X}, \mathbf{P})$.
- $\mathsf{Verify}(PK_{\mathcal{T}}, BB, L, n_C, \mathbf{X}, \mathbf{P}, k)$. Parse $PK_{\mathcal{T}}$ as a vector $(pk_T, \mathfrak{m}, \rho)$, $\mathbf{X}$ as a vector of length n_C, and $\mathbf{P}$ as a vector $(\mathbf{P_{dupl}}, \mathbf{C_1}, P_{mix,1}, \mathbf{C_2}, P_{mix,2}, \mathbf{C_3}, P_{mix,3}, \mathbf{P_{inelig}}, \mathbf{P_{dec}})$, outputting 0 if parsing fails or $\mathsf{VerKey}((k, pk_T, \mathfrak{m}), \rho, k) \neq 1$. Let $m_C = |\mathfrak{m}|$. If $n_C > m_C$, then output 0. Otherwise, perform the following checks:
  1) *Check removal of invalid ballots:* Compute $\{b_1, \dots, b_\ell\}$ as per Step (1) of the tallying algorithm. If $\{b_1, \dots, b_\ell\} = \emptyset$ and $\mathbf{X}$ is a zero-filled vector, then output 1. Otherwise, proceed as follows.
  2) *Check duplicate elimination:* Check that $\mathbf{P_{dupl}}$ is a vector of length ℓ and that for all $1 \le i \le \ell$, either: i) $|\mathbf{P_{dupl}}[i]| = 1$ and there exists $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$ such that $\mathsf{VerPET}((pk_T, b_i[2], b_j[2], 1), \mathbf{P_{dupl}}[i][1], k) = 1$, or ii) $|\mathbf{P_{dupl}}[i]| = \ell - 1$ and for all $j \in \{1, \dots, i-1, i+1, \dots, \ell\}$ we have $\mathsf{VerPET}((pk_T, b_i[2], b_j[2], 0), \mathbf{P_{dupl}}[i][j], k) = 1$.
  3) *Check mixing:* Compute $\mathbf{BB}$ as per Step (2) of the tallying algorithm, suppose $\mathbf{BB} = (b'_1, \dots, b'_{|\mathbf{BB}|})$, and check that $\mathsf{VerMix}((pk_T, (b'_1[1], \dots, b'_{|\mathbf{BB}|}[1]), \mathbf{C_1}), P_{mix,1}, k) = 1 \wedge \mathsf{VerMix}((pk_T, (b'_1[2], \dots, b'_{|\mathbf{BB}|}[2]), \mathbf{C_2}), P_{mix,2}, k) = 1 \wedge \mathsf{VerMix}((pk_T, L, \mathbf{C_3}), P_{mix,3}, k) = 1$.
  4) *Check removal of ineligible ballots:* Check that $\mathbf{P_{inelig}}$ is a vector of length $|\mathbf{C_2}|$ and that for all $1 \le i \le |\mathbf{C_2}|$, either: i) $|\mathbf{P_{inelig}}[i]| = 1$ and there exists $c \in \mathbf{C_3}$ such that $\mathsf{VerPET}((pk_T, \mathbf{C_2}[i], c, 1), \mathbf{P_{inelig}}[i][1], k) = 1$, or ii) $|\mathbf{P_{inelig}}[i]| = |\mathbf{C_3}|$ and for all $1 \le j \le |\mathbf{C_3}|$ we have $\mathsf{VerPET}((pk_T, \mathbf{C_2}[i], \mathbf{C_3}[j], 0), \mathbf{P_{inelig}}[i][j], k) = 1$.
  5) *Check decryption:* Compute $\mathbf{C'_1}$ as follows: $\mathbf{C'_1} \leftarrow ()$; **for** $1 \le i \le |\mathbf{C_1}| \wedge |\mathbf{P_{inelig}}[i]| = 1$ **do** $\mathbf{C'_1} \leftarrow \mathbf{C'_1} \parallel (\mathbf{C_1}[i])$. Check that there exists $\beta_1, \dots, \beta_{|\mathbf{C'_1}|}$ such that $\mathbf{X}[i] = |\{j : 1 \le j \le |\mathbf{C'_1}| \wedge \beta_j = i\}|$ and for all $1 \le i \le |\mathbf{C'_1}|$ we have $\mathsf{VerDec}((pk_T, \mathbf{C'_1}[i], \beta_i), \mathbf{P_{dec}}[i], k) = 1$.

  Output 0 if any of the above checks do not hold. Otherwise, if all the above checks succeed, output 1.

since JCJ ensures that n_B is bounded by a polynomial in k and the secret keys are just random nonces. Now it is easy to see that the only step of Tally that we need to be concerned about is the step in which duplicate ballots are removed. This is because the checks performed in the other steps all pass with overwhelming probability when the computation is done honestly. In the step to remove duplicate ballots, we need to make sure that there are not multiple ballots computed using sk_{n+1}. As we argued above, sk_{n+1} is unique among the secret keys, so the ballot computed using sk_{n+1} will not be removed, and we will get that $\mathbf{X} = \mathbf{Y}$. Therefore, JCJ satisfies correctness. □

Lemma 25. *Suppose* Γ, Σ_1, Σ_2, Σ_3, $\Sigma 4$, Σ_5, Σ_6 *and* $\mathcal{H}$ *satisfy the preconditions of Figure 3. We have* $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$ *satisfies Completeness.*

Proof. Let $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H}) = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$, $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$, $\mathsf{FS}(\Sigma_4, \mathcal{H}) = (\mathsf{ProveDec}, \mathsf{VerDec})$, $\mathsf{FS}(\Sigma_5, \mathcal{H}) = (\mathsf{ProvePET}, \mathsf{VerPET})$, and $\mathsf{FS}(\Sigma_6, \mathcal{H}) = (\mathsf{ProveMix}, \mathsf{VerMix})$. Suppose k is a security parameter, BB is a bulletin board, and n_C is an integer. Further suppose $(PK_\mathcal{T}, SK_\mathcal{T})$ is a key pair, m_B and m_C are integers, L is an electoral roll (i.e., a set of public keys output by Register), and $(\mathbf{X}, P)$ is a tally, such that $(PK_\mathcal{T}, SK_\mathcal{T}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$ and $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_\mathcal{T}, BB, L, n_C, k)$. Moreover, suppose $n_C \le m_C$. By definition of Setup, there exist coins r such that $(pk, sk, \mathfrak{m}) = \mathsf{Gen}(k; r)$, $PK_\mathcal{T} \leftarrow (pk, \mathfrak{m}, \rho)$, $SK_\mathcal{T} \leftarrow (pk, sk)$ and $m_C = |\mathfrak{m}|$, where ρ is an output of $\mathsf{ProveKey}((k, PK_\mathcal{T}, \mathfrak{m}), (SK_\mathcal{T}, r), k)$. Since n_C is at most $|\mathfrak{m}|$, we have that any $\beta \in \{1, \ldots, n_C\}$ is in Γ's message space. Moreover, by the definition of Tally, vector $\mathbf{X}$ is of length n_C and P is a vector $(\mathbf{P_{dupl}}, \mathbf{C_1}, P_{mix,1}, \mathbf{C_2}, P_{mix,2}, \mathbf{C_3}, P_{mix,3}, \mathbf{P_{inelig}}, \mathbf{P_{dec}})$. It follows that Verify can parse P and $\mathbf{X}$ successfully. Moreover, by completeness of $(\mathsf{ProveKey}, \mathsf{VerKey})$, we have $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) = 1$ with overwhelming probability. Suppose $\{b_1, \ldots, b_l\}$ is the largest subset of BB satisfying the conditions given by algorithm Tally. If $\{b_1, \ldots, b_l\} = \emptyset$, then $\mathbf{X}$ is a zero-filled vector and Verify accepts, concluding our proof. Otherwise, we proceed by showing that checks (2)–(5) of Verify succeed:

- *Check duplicate elimination.* The check succeeds by completeness of $(\mathsf{ProvePET}, \mathsf{VerPET})$, namely, for all $1 \le i \le \ell$ we have either: i) $|\mathbf{P_{dupl}}[i]| = 1$ and there exists $j \in \{1, \ldots, i-1, i+1, \ldots, \ell\}$ such that $\mathsf{VerPET}((PK_\mathcal{T}, b_i[2], b_j[2], 1), \mathbf{P_{dupl}}[i][1], k) = 1$; or ii) $|\mathbf{P_{dupl}}[i]| = \ell - 1$ and for all $j \in \{1, \ldots, i-1, i+1, \ldots, \ell\}$ we have $\mathsf{VerPET}((PK_\mathcal{T}, b_i[2], b_j[2], 0)), \mathbf{P_{dupl}}[i][j], k) = 1$.
- *Check mixing.* Suppose $BB = (b'_1, \ldots, b'_{|BB|})$. Then by the completeness of $(\mathsf{ProveMix}, \mathsf{VerMix})$, we have that $\mathsf{VerMix}((PK_\mathcal{T}, (b'_1[1], \ldots, b'_{|\mathbf{BB}|}[1]), \mathbf{C_1}), P_{mix,1}, k) = 1 \wedge \mathsf{VerMix}((PK_\mathcal{T}, (b'_1[2], \ldots, b'_{|\mathbf{BB}|}[2]), \mathbf{C_2}), P_{mix,2}, k) = 1 \wedge \mathsf{VerMix}((PK_\mathcal{T}, L, \mathbf{C_3}), P_{mix,3}, k) = 1$.
- *Check removal of ineligible ballots.* By Step (4) of Tally, we have $\mathbf{P_{inelig}}$ is a vector of length $|\mathbf{C_2}|$. Moreover, by completeness of $(\mathsf{ProvePET}, \mathsf{VerPET})$, for all $1 \le i \le |\mathbf{C_2}|$ we have either: i) $|\mathbf{P_{inelig}}[i]| = 1$ and there exists $c \in \mathbf{C_3}$ such that $\mathsf{VerPET}((PK_\mathcal{T}, \mathbf{C_2}[i], c, 1), \mathbf{P_{inelig}}[i][1], k) = 1$; or ii) $|\mathbf{P_{inelig}}[i]| = |\mathbf{C_3}|$ and for all $1 \le j \le |\mathbf{C_3}|$ we have $\mathsf{VerPET}((PK_\mathcal{T}, \mathbf{C_2}[i], \mathbf{C_3}[j], 0), \mathbf{P_{inelig}}[i][j], k) = 1$. It follows that the check succeeds.
- *Check decryption.* Verify computes the set $\mathbf{C'_1}$ such that it includes only elements c_i of $\mathbf{C_1}$ for which $|\mathbf{P_{inelig}}[i]| = 1$. Then, by the definition of Tally and the completeness of $(\mathsf{ProveDec}, \mathsf{VerDec})$, we have that $\mathsf{VerDec}((PK_\mathcal{T}, \mathbf{C'_1}[\mathbf{i}], \beta_i), \mathbf{P}[9][i], k) = 1$ for all $1 \le i \le |\mathbf{C'_1}|$. Furthermore, in step 5 of Tally, ballots $\mathbf{C_1}[i]$ are only counted for a candidate when $1 \le i \le |\mathbf{C_1}| \wedge |\mathbf{P_{inelig}}[i]| = 1$, which is exactly how $\mathbf{C'_1}$ is defined. Therefore, there exists $\beta_1, \ldots \beta_{|\mathbf{C'_1}|}$ such that $\mathbf{X}[i] = |\{j : 1 \le j \le |\mathbf{C'_1}| \wedge \beta_j = i\}|$.

It follows that all the required checks succeed and Verify outputs 1, concluding our proof. □

Lemma 26. *Suppose* Γ, Σ_1, Σ_2, Σ_3, $\Sigma 4$, Σ_5, Σ_6 *and* $\mathcal{H}$ *satisfy the preconditions of Figure 3. Further suppose* Γ *is collision-free. We have* $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$ *satisfies Injectivity.*

The proof of Lemma 26 is similar to the proof of Lemma 17.

Proof sketch. Generalized JCJ ballots contain encrypted choices, hence, collision-freeness of the encryption scheme ensures that distinct choices are not mapped to the same ballot. □

Generalized JCJ can be instantiate to derive JCJ:

Definition 29 (JCJ). JCJ *[83] is* $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$*, where* Γ *is a modified version of El Gamal [58] invented by Juels et al. [83, §4] that can be seen as a simplified version of Cramer–Shoup [49],* Σ_1 *is the proof of key construction by Gennaro et al. [62],* Σ_4 *is the conjunction [46] of two Schnorr proofs [109],* Σ_5 *is the PET by MacKenzie et al. [96],* Σ_6 *is either the mixnet of Furukawa and Sako [61] or Neff [99], and* $\mathcal{H}$ *is a random oracle. Juels et al. leave* Σ_2 *and* Σ_3 *unspecified.*

Juels et al. [83] do not mandate particular cryptographic primitives, so Definition 29 might be seen more as an instantiation of their scheme than an exact recollection of it. We assume that the primitives in Definition 29 satisfy the properties required by generalized JCJ. We also assume that the sigma protocols satisfy special soundness and special honest verifier zero-knowledge, hence, Theorem 12 is applicable.

To show that JCJ is an election scheme, we must demonstrate that Correctness, Completeness and Injectivity are satisfied. Correctness follows immediately from Lemma 24 and Completeness follows from Lemma 25. We show that Injectivity is also satisfied.

A non-interactive proof system derived from a sigma protocol for proving correct key construction is sufficient to ensure that El Gamal is collision-free:

Lemma 27. *Suppose Σ_1 is a sigma protocol that proves correct key construction and $\mathcal{H}$ is a hash function. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$. *Further suppose for all security parameters k, public keys pk, message spaces $\mathfrak{m}$ and proofs ρ, we have* $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) = 1$ *implies $h \neq 0$, there exists p, q, g and h such that $pk = (p, q, g, h)$ and (p, q, g) are cryptographic parameters, and $\mathfrak{m} = \{1, \ldots, p-1\}$. It follows that multiplicatively homomorphic El Gamal is collision-free for $m_1, m_2 \in \mathfrak{m}$.*

The proof of Lemma 27 is similar to the proof of Lemma 19.

Proof. Suppose k is a security parameter, pk is a public key, $\mathfrak{m}$ is a message space, ρ is a proof, $m_1, m_2 \in \mathfrak{m}$ are messages and r_1 and r_2 are coins such that $\mathsf{VerKey}((k, pk, \mathfrak{m}), \rho, k) = 1$, $m_1 \neq m_2 \vee r_1 \neq r_2$, $\mathfrak{m} = \{1, \ldots, p-1\}$, and $pk = (p, q, g, h)$ for some p, q, g and h. Further suppose that c_1 and c_2 are ciphertexts such that $c_1 = \mathsf{Enc}(pk, m_1; r_1)$, $c_2 = \mathsf{Enc}(pk, m_2; r_2)$, and Enc is El Gamal's encryption algorithm. If $r_1 \neq r_2$, then we proceed as follows. By definition of Enc, we have $c_1[1] = g^{r_1} \pmod p$ and $c_2[1] = g^{r_2} \pmod p$. Since r_1 and r_2 are distinct, we have $g^{r_1} \not\equiv g^{r_2} \pmod p$. (We implicitly assume that coins r_1 and r_2 are selected from the coin space $\mathbb{Z}_q^*$, hence, $g^{r_1} = g^{r_1} \bmod p$ and $g^{r_2} = g^{r_2} \bmod p$.) It follows that $c_1 \neq c_2$. Otherwise $(r_1 = r_2)$, we have $m_1 \neq m_2$ and we proceed as follows. By definition of Enc, we have $c_1[2] = h^{r_1} \cdot m_1 \pmod p$ and $c_2[2] = h^{r_2} \cdot m_2 \pmod p$. Since $h \neq 0$, we have $h^{r_1} \cdot m_1 \not\equiv h^{r_1} \cdot m_2 \pmod p$. □

Given that ciphertexts generated by the modified version of El Gamal used in JCJ [83, §4] encapsulate El Gamal ciphertexts, the proof of key construction by Gennaro et al. [62] is sufficient to ensure that El Gamal is collision-free:

Corollary 28. *The modified version of El Gamal used in JCJ [83, §4] is collision-free its message space $\mathfrak{m}$.*

The sigma protocol for proving correct key construction by Gennaro et al. [62] does not explicitly require the suitability of cryptographic parameters to be checked, hence, Lemma 27 is not immediately applicable. Nonetheless, we can trivially make the necessary checks explicit and, hence, the non-interactive proof system derived from the sigma protocol for proving correct key construction by Gennaro et al. is sufficient to ensure that El Gamal is collision-free. It follows that JCJ satisfies Injectivity, hence, JCJ is an election scheme.

APPENDIX I
PROOF: JCJ IS VERIFIABLE

Elections schemes constructed from generalized JCJ satisfy individual (§I-A), universal (§I-B) and eligibility (§I-C) verifiability, hence, such schemes satisfy election verifiability with internal authentication (§I-D). It follows that JCJ satisfies election verifiability (§I-E).

A. Individual verifiability

Proposition 29. *Suppose Γ, Σ_1, Σ_2, Σ_3, Σ_4, Σ_5, Σ_6 and $\mathcal{H}$ satisfy the preconditions of Figure 3. Further suppose that Γ is collision-free for its message space $\mathfrak{m}$ and Σ_1 satisfies special soundness and special honest verifier zero-knowledge. We have* $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$ *satisfies individual verifiability.*

Proof. Let $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$=$(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ and $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$. Suppose k is a security parameter, $PK_{\mathcal{T}}$ is a public key, n_C is an integer, and β and β' are choices. Further suppose that (pk, sk) and (pk', sk') are key pairs and b and b' are ballots such that $(pk, sk) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$, $(pk', sk') \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$, $b \leftarrow \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k)$, $b' \leftarrow \mathsf{Vote}(sk', PK_{\mathcal{T}}, n_C, \beta', k)$, $b \neq \perp$, and $b' \neq \perp$. By definition of Vote, we have $PK_{\mathcal{T}}$ is a vector $(pk_{\mathcal{T}}, \mathfrak{m}, \rho)$ and $\mathsf{VerKey}((k, pk_{\mathcal{T}}, \mathfrak{m}), \rho, k) = 1$. By definition of Vote, $b[2]$ and $b'[2]$ are ciphertexts such that $b[2] \leftarrow \mathsf{Enc}(pk_{\mathcal{T}}, sk)$ and $b'[2] \leftarrow \mathsf{Enc}(pk_{\mathcal{T}}, sk')$, where $sk, sk' \in \mathfrak{m}$. Furthermore, the ciphertexts are constructed using random coins—i.e., the coins used by $b[2]$ and $b'[2]$ will be distinct with overwhelming probability. Since Γ is collision-free for $\mathfrak{m}$, we have $b[2] \neq b'[2]$ and $b \neq b'$ with overwhelming probability, concluding our proof. □

B. Universal verifiability.

Proposition 30. *Suppose Γ is a homomorphic asymmetric encryption scheme, Σ_1, Σ_2, Σ_3, Σ_4, Σ_5 and Σ_6, are sigma protocols and $\mathcal{H}$ is a hash function such that the conditions of Figure 3 are satisfied. Further suppose that Γ satisfies IND-CPA and Σ_1 and Σ_6 satisfy special soundness and special honest verifier zero-knowledge. We have* $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$ *satisfies universal verifiability.*

The proof is similar in structure to the universal verifiability proof for Helios (§F-B): we use the definition of the verification algorithm to construct the tally $\mathbf{X}$ given by the adversary, and then show that $\mathbf{X}$ is equal to the correct tally.

Proof. Suppose that an execution of $\mathsf{Exp\text{-}UV\text{-}Int}(\Pi, \mathcal{A}, k)$ computes

$$
\begin{aligned}
&(PK_{\mathcal{T}}, n_V) \leftarrow \mathcal{A}(k);\\
&\textbf{for } 1 \leq i \leq n_V \textbf{ do } (pk_i, sk_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)\\
&L \leftarrow \{pk_1, \ldots, pk_{n_V}\};\\
&M \leftarrow \{(pk_1, sk_1), \ldots, (pk_{n_V}, sk_{n_V})\};\\
&(BB, n_C, \mathbf{X}, P) \leftarrow \mathcal{A}(M);\\
&\mathbf{Y} \leftarrow \textit{correct-tally}(PK_{\mathcal{T}}, BB, M, n_C, k);
\end{aligned}
$$

such that $\mathsf{Verify}(PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P, k) = 1$. The JCJ verification algorithm checks the proof ρ in $PK_{\mathcal{T}} = (pk_{\mathcal{T}}, \mathfrak{m}, \rho)$, so $\mathsf{VerKey}((k, pk_{\mathcal{T}}, \mathfrak{m}), \rho, k) = 1$ and by simulation sound extractability we are assured that pk_T was honestly generated—i.e., there exists r and $SK_{\mathcal{T}}$ such that $(pk_T, SK_{\mathcal{T}}, \mathfrak{m}) = \mathsf{Gen}(k; r)$. We now look at each step in the Verify algorithm.

- *Check removal of invalid ballots:* Let $\{b_1, \ldots, b_\ell\}$ be the largest subset of BB such that for all $1 \leq i \leq \ell$ we have b_i is a vector of length 4 and $\mathsf{VerCiph}((pk_T, b_i[1]\{1,\ldots,n_C\}), b_i[3], k) = 1 \wedge \mathsf{VerBind}((pk_T,\ b_i[1], b_i[2]), b_i[4], k) = 1$. If this set is empty, then Verify would only accept if $\mathbf{X}[i] = 0$ for all $1 \le i \le n_C$ and $\mathbf{P} = \perp$. Since the set is empty, no ballots b were posted to the bulletin board for which $\mathsf{VerCiph}((pk_T, b_i[1], \{1,\ldots,n_C\}), b_i[3], k) = 1 \wedge \mathsf{VerBind}((pk_T, b_i[1], b_i[2]), b_i[4], k) = 1$. By the completeness of the non-interactive proof system, if the ballots were outputs of the Vote function, then they would verify. Therefore, no ballots on the bulletin board were the output of the Vote function, so we will have that $\mathbf{Y}$ is also a vector of zeroes. Thus we would have that $\mathbf{X} = \mathbf{Y}$ and conclude our proof. Now let's assume that $\{b_1,\ldots,b_\ell\} \neq \emptyset$.

We must have for all choices $\beta \in \{1,\ldots,n_C\}$, secret keys sk such that $(pk, sk) \in M$, coins r, and ballots $b = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k; r)$ that $b \notin BB\backslash\{b_1,\ldots,b_\ell\}$ with overwhelming probability, since otherwise we would have a contradiction: $\{b_1,\ldots,b_\ell\}$ is not the largest subset of BB satisfying the conditions of the Tally algorithm. Therefore, we must have that

$$\begin{aligned} &\mathit{correct\text{-}tally}(PK_{\mathcal{T}}, M, BB, n_C, k) \\ &\quad = \mathit{correct\text{-}tally}(PK_{\mathcal{T}}, M, \{b_1,\ldots,b_\ell\}, n_C, k) \end{aligned} \tag{5}$$

- *Check duplicate elimination:* Next, the verification algorithm checks that duplicate votes were properly eliminated—i.e., that either $|\mathbf{P_{dupl}}[i]| = 1 \wedge \exists j \in \{1,\ldots,i-1,i+1,\ldots,\ell\}$ such that $\mathsf{VerPET}((pk_T, b_i[2], b_j[2]), \mathbf{P_{dupl}}[i][1], 1, k) = 1$ or $|\mathbf{P_{dupl}}[i]| = \ell - 1 \wedge \forall j \in \{1,\ldots,i-1,i+1,\ldots, n\}$ such that $\mathsf{VerPET}((pk_T, b_i[2], b_j[2], 0), \mathbf{P_{dupl}}[i][j], k) = 1$. Let $\mathbf{BB}$ be constructed as in Step (2) of the JCJ tallying algorithm. By the simulation sound extractability of the $\mathbf{P_{dupl}}[i]$, we are assured that there are no duplicate votes in $\mathbf{BB}$.
- *Check mixing:* Now the ballots in $\mathbf{BB}$ are permuted and re-encrypted using a mixnet. While permuting the ballots isn't necessary for verifiability, the associated proofs are necessary because they show that the re-encryption was done properly (for example, they ensure that the encrypted ballot was multiplied by an encryption of the identity element, and not some other group element that might change the vote). Let C_1 denote the list of mixed re-encryptions of candidates, C_2 denote the list of mixed re-encryptions of voters' secret keys from the ballots, and C_3 denote the mixed list of encryptions of voters' secret keys. The permutation used to generate C_3 is different from the permutation used to generate C_1 and C_2, but this isn't important to the verifiability of the scheme. We have that $\mathsf{VerMix}((pk_T, (b'_1[1],\ldots,b'_{|\mathbf{BB}|}[1]), \mathbf{C_1}), P_{mix,1}, k) = 1 \wedge \mathsf{VerMix}((pk_T, (b'_1[2],\ldots,b'_{|\mathbf{BB}|}[2]), \mathbf{C_2}), P_{mix,2}, k) = 1 \wedge \mathsf{VerMix}((pk_T, L, \mathbf{C_3}), P_{mix,3}, k) = 1$. By simulation sound extractability, we have that each C_i does indeed contain re-encryptions of the original lists in $\mathbf{BB}$.
- *Check removal of ineligible ballots:* Next, the verification algorithm ensures that ineligible ballots are removed properly. The verification algorithm checks that each PET in $\mathbf{P}[8] = \mathbf{P_{inelig}}$ is valid. Let $C'_1 \subseteq C_1$ be the set of $C_1[i] \in C_1$ for which $|P_{inelig}| = 1$ and there exists $c \in C_3$ such that $\mathsf{VerPET}((pk_T, \mathbf{C_2}[i], c, 1), \mathbf{P_{inelig}}[i][1], k) = 1$. In other words, C'_1 is the set of encryptions of candidates generated using a valid voter's secret key.
- *Check decryption:* Finally, the verification algorithm checks the proofs that all of the ballots in C'_1 are properly decrypted. The verification algorithm outputs 1, so by simulation sound extractability we are assured that the multiset of candidates given by decrypting the ballots in C'_1 is correct. We will call this multiset C_{Final}. Finally the verification algorithm checks that this multiset corresponds to the vector $\mathbf{X}$.

We can see that C_{Final} satisfies the following properties. First, every element β in C_{Final} corresponds to a ballot $b \in BB$ which was generated using Vote with a valid voter's secret key. This is ensured by steps (1), (3), and (4) of the verification algorithm. Second, for every $\beta \in C_{Final}$, the ballot corresponding to this β was the only one constructed under its particular secret key—i.e., $\neg\exists b', \beta', r' : b' \in BB \setminus \{b\} \wedge b' = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta', k; r')\}$, where b is the ballot corresponding to β. This is ensured by steps (2) and (3) of the verification algorithm. Therefore, we have that each $\beta \in C_{Final}$ corresponds to a ballot in $authorized(PK_{\mathcal{T}}, BB, M, n_C, k)$. Finally $\mathbf{X}[\beta] = k$ iff $\exists^{=k}\beta \in C_{Final}$. This is ensured by step (5) of the verification algorithm. Since these are the exact properties that define $\mathit{correct\text{-}tally}(PK_{\mathcal{T}}, M, \{b_1,\ldots,b_\ell\}, n_C, k)$, we must have that $\mathbf{X} = \mathbf{Y}$. $\square$

C. Eligibility Verifiability

We proceed as follows. First, we derive an IND-1-CPA encryption scheme from generalized JCJ (§I-C1). Secondly, we introduce an experiment that is equivalent to Exp-EV-Int-Weak for JCJ (§I-C2). Finally, we prove that JCJ satisfies our new experiment (§I-C3), using the IND-1-CPA encryption scheme.

1) Encryption scheme from generalized JCJ:

Definition 30. *Suppose* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is an asymmetric encryption scheme,* Σ_1 *proves correct key construction,* Σ_3 *proves conjunctive plaintext knowledge, and* $\mathcal{H}$ *is a random oracle. Let* $\mathsf{FS}(\Sigma_1, \mathcal{H}) = (\mathsf{ProveKey}, \mathsf{VerKey})$ *and* $\mathsf{FS}(\Sigma_3, \mathcal{H}) = (\mathsf{ProveBind}, \mathsf{VerBind})$. *We define* $\Pi_{JCJ}(\Pi, \Sigma_1, \Sigma_3, \mathcal{H}) = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *as follows:*

- $\mathsf{Gen}'(k; r)$: *Compute* $(pk_T, sk_T, \mathfrak{m}) \leftarrow \mathsf{Gen}(k; r)$; $\rho \leftarrow \mathsf{ProveKey}((k, PK_{\mathcal{T}}, \mathfrak{m}), (SK_{\mathcal{T}}, r), k)$; $PK_{\mathcal{T}} \leftarrow (pk_T, \mathfrak{m}, \rho)$; $SK_{\mathcal{T}} \leftarrow ((PK_{\mathcal{T}}, k), sk_T)$; $\mathfrak{m}' \leftarrow \{(m_1, m_2) \mid m_1, m_2 \in \mathfrak{m}\}$. *Output* $((PK_{\mathcal{T}}, k), SK_{\mathcal{T}}, \mathfrak{m}')$.
- $\mathsf{Enc}'(pk, m)$: *Parse* m *as a vector* (β, d), pk *as a vector* $(PK_{\mathcal{T}}, k)$, *and* $PK_{\mathcal{T}}$ *as a vector* $(pk_T, \mathfrak{m}, \rho)$, *outputting* $\perp$ *if parsing fails. Select coins* r_1 *and* r_2 *and*

compute $c_1 \leftarrow \mathsf{Enc}(pk_T, \beta; r_1)$*;* $c_2 \leftarrow \mathsf{Enc}(pk_T, d; r_2)$*;* $\tau \leftarrow \mathsf{ProveBind}((pk_T, c_1, c_2), (\beta, r_1, d, r_2), k)$*. Output* (c_1, c_2, τ).

- $\mathsf{Dec}'(SK_{\mathcal{T}}, c)$ *: Parse* c *as* (c_1, c_2, τ)*,* $SK_{\mathcal{T}}$ *as* (pk, sk)*,* pk *as* $(PK_{\mathcal{T}}, k)$*, and* $PK_{\mathcal{T}}$ *as* $(pk_T, \mathfrak{m}, \rho)$*, outputting* $\perp$ *if parsing fails or* $\mathsf{VerBind}((pk_T, c_1, c_2), \tau, k) \neq 1$*. Compute* $\beta \leftarrow \mathsf{Dec}(sk, c_1); d \leftarrow \mathsf{Dec}(sk, c_2)$ *and output* (β, d).

The key generation algorithm Gen' outputs a public key $(PK_{\mathcal{T}}, k)$, where $PK_{\mathcal{T}} = (pk_T, \mathfrak{m}, \rho)$. Parameters $\mathfrak{m}$, ρ, and k are used in our proof of eligibility verifiability, but are not required by the encryption scheme.

Proposition 31. $\Pi_{JCJ}(\Pi, \Sigma_1, \Sigma_3, \mathcal{H})$ *is an asymmetric encryption scheme satisfying IND-1-CPA, where* Π*,* Σ_1*,* Σ_3 *and* $\mathcal{H}$ *satisfy the preconditions of Definition 30.*

Proof. The proof that this scheme satisfies IND-1-CPA is adapted from that of [22, Theorem 5.1]. We will show that if there is an adversary $\mathcal{A}'$ that can win the IND-1-CPA game for the scheme, then there is another adversary $\mathcal{A}$ that can win the IND-CPA game for the following scheme: Let $\Gamma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an asymmetric encryption scheme satisfying IND-CPA. Define $\Gamma' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- $\mathsf{Gen}'(k; r)$: Compute $(pk, sk, \mathfrak{m}) \leftarrow \mathsf{Gen}(k; r)$; $\rho \leftarrow \mathsf{ProveKey}((k, pk, \mathfrak{m}), (sk, r), k)$; $PK_{\mathcal{T}} \leftarrow (pk, \mathfrak{m}, \rho)$; $pk' = (PK_{\mathcal{T}}, k)$; $SK_{\mathcal{T}} \leftarrow (pk', sk)$; $\mathfrak{m}' \leftarrow \{(m_1, m_2) \mid m_1, m_2 \in \mathfrak{m}\}$, and output $(pk', SK_{\mathcal{T}}, \mathfrak{m}')$.
- $\mathsf{Enc}'(pk, m)$: Parse m as a vector (m_0, m_1), pk as $(PK_{\mathcal{T}}, k)$, and $PK_{\mathcal{T}}$ as $(pk', \mathfrak{m}, \rho)$, outputting $\perp$ if parsing fails. Compute $c_0 \leftarrow \mathsf{Enc}(pk', m_0)$; $c_1 \leftarrow \mathsf{Enc}(pk', m_1)$, and output (c_0, c_1).
- $\mathsf{Dec}'(sk, c)$: Parse c as a vector (c_0, c_1), pk as $(PK_{\mathcal{T}}, k)$, and $PK_{\mathcal{T}}$ as $(pk', \mathfrak{m}, \rho)$, outputting $\perp$ if parsing fails. Compute $m_0 \leftarrow \mathsf{Dec}(sk, c_0)$; $m_1 \leftarrow \mathsf{Dec}(sk, c_1)$, and output (m_0, m_1).

It is straightforward to see that this scheme satisfies IND-CPA.

Now we begin the reduction. Let $\mathcal{A}'$ be an adversary that wins the IND-1-CPA game against $\Pi_{JCJ}(\Pi, \Sigma_1, \Sigma_3, \mathcal{H})$ with non-negligible probability. We will construct an adversary $\mathcal{A}$ that wins the IND-CPA game against the Γ' defined above with non-negligible probability. $\mathcal{A}$ is first given a public key pk, where $pk = (PK_{\mathcal{T}}, k)$ and $PK_{\mathcal{T}} = (pk', \mathfrak{m}, \rho)$. $\mathcal{A}$ forwards pk to $\mathcal{A}'$. $\mathcal{A}'$ may make queries to its random oracle. $\mathcal{A}$ will simulate the random oracle and keep a list $\mathcal{H}$ of all previously asked queries. If $\mathcal{A}'$ makes a query for a value already in $\mathcal{H}$, $\mathcal{A}$ responds with a value consistent with the list. If $\mathcal{A}'$ makes a query for a new value, $\mathcal{A}$ chooses a value uniformly at random from the range of the random oracle and adds the query/response pair to $\mathcal{H}$. We will denote by $\mathcal{H}(x)$ the response y such that (x, y) is in $\mathcal{H}$, and $\perp$ if no such query/response pair is in $\mathcal{H}$.

Next $\mathcal{A}'$ will output two messages m_0, m_1 of the form (β, d). $\mathcal{A}$ outputs m_0, m_1 and receives a challenge ciphertext $c^* = (c_0^*, c_1^*)$. $\mathcal{A}$ then picks a challenge $chal^*$ at random from the challenge space. In order to generate the proof of conjunctive plaintext knowledge that $\mathcal{A}'$ expects, $\mathcal{A}$ will use the simulator Sim for the sigma protocol associated with $\mathsf{ProveBind}$. This simulator exists due to the special honest verifier zero-knowledge property of the sigma protocol. $\mathcal{A}$ runs $Sim((pk', c_0^*, c_1^*), chal^*)$ to obtain the simulated proof $\tau^* = (comm^*, resp^*)$, and adds the pair $((pk', c_0^*, c_1^*)||comm^*, chal^*)$ to $\mathcal{H}$. If there is already an entry corresponding to the query $(pk', c_0^*, c_1^*)||comm^*$ in $\mathcal{H}$, $\mathcal{A}$ aborts with "Error 1". $\mathcal{A}$ then gives (c_0^*, c_1^*, τ^*) to $\mathcal{A}'$.

$\mathcal{A}'$ will next output its vector of decryption queries $\mathbf{c}$. Let $|\mathbf{c}| = \ell$. For each $i \in \{1, \ldots, \ell\}$, $\mathcal{A}$ will obtain the response to the query $\mathbf{c}[i]$ using the following procedure. First, $\mathcal{A}$ checks that $\mathbf{c}[i]$ is a valid ciphertext, i.e, that $\mathbf{c}[i] = (c_i^0, c_i^1, \tau_i)$ where $\tau_i = (comm_i, resp_i)$ such that $\mathsf{VerBind}((pk', c_i^0, c_i^1), (comm_i, \mathcal{H}((pk', c_i^0, c_i^1)||comm_i)), resp_i, k) = 1$. If there is no entry $(x, y) \in \mathcal{H}$ such that $x = (pk', c_i^0, c_i^1)||comm_i$, $\mathcal{A}$ adds it as if $\mathcal{A}'$ had queried its random oracle on that value. If these conditions do not hold or $\mathbf{c}[i] = (c_0^*, c_1^*, \tau^*)$, the response for $\mathbf{c}[i]$ will be $\perp$. Now $\mathcal{A}$ checks to see where $\mathcal{A}'$ queried on $(pk', c_i^0, c_i^1)||comm_i$. If $\mathcal{A}'$ never made such a query, $\mathcal{A}$ aborts with "Error 2". $\mathcal{A}$ simulates a new copy of $\mathcal{A}'$ up to the point of that query, but this time responds with a new, uniformly random value. All other queries are answered as they were in the "main" run of $\mathcal{A}'$. $\mathcal{A}$ continues the simulation until $\mathcal{A}'$ outputs $\mathbf{c}'$. If $\mathbf{c}'$ contains an entry (c_j^0, c_j^1, τ_j) such that $c_j^0 = c_i^0, c_j^1 = c_i^1$ and $comm_j = comm_i$, then $\mathcal{A}$ uses the special soundness extractor for the sigma protocol to obtain the witness w_i for the statement. This witness consists of the messages and random coins used to generate the ciphertexts. $\mathcal{A}$ uses this witness to answer the decryption query in the "main" run. Finally, $\mathcal{A}'$, will output a bit b, which $\mathcal{A}$ outputs as well.

The remainder of the proof is almost exactly the same as that of [22, Theorem 5.1], and so is omitted here. □

2) Variant of $\mathsf{Exp\text{-}EV\text{-}Int\text{-}Weak}$*:*

$\mathsf{Exp\text{-}EV\text{-}1\text{-}Int}(\Pi, \mathcal{A}, k) =$
$(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$;
$(pk, sk) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$;
$Rvld \leftarrow \emptyset$;
$(n_C, \beta, b) \leftarrow \mathcal{A}^R(PK_{\mathcal{T}}, pk, k)$;
if $\exists r : b = \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k; r) \wedge b \neq \perp$ $\wedge\, b \notin Rvld$ **then**
  return 1
else
  return 0

Lemma 32. *Let* Π *be Generalized JCJ, where the encryption scheme* Γ *satisfies IND-CPA. Then we have*

$$\forall \mathcal{A}\ \exists \mu\ \forall k\ .\ \mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}1\text{-}Int}(\Pi, \mathcal{A}, k)) \leq \mu(k)$$
$$\Leftrightarrow \forall \mathcal{A}'\ \exists \mu'\ \forall k'\ .\ \mathsf{Succ}(\mathsf{Exp\text{-}EV\text{-}Int\text{-}Weak}(\Pi, \mathcal{A}', k')) \leq \mu'(k'),$$

where $\mathcal{A}$ *and* $\mathcal{A}'$ *are PPT adversaries,* μ *and* μ' *are negligible functions, and* k *and* k' *are security parameters.*

The forward implication is required by Proposition 33 and we provide a formal proof below. A proof of the reverse implication is straight-forward and we omit our formal proof.

Proof. We will show that if an adversary wins Exp-EV-Int-Weak, then there exists an adversary that wins Exp-EV-1-Int. Let $\mathcal{A}'$ be the adversary that wins Exp-EV-Int-Weak with non-negligible probability. We will construct the adversary $\mathcal{A}$ for Exp-EV-1-Int. The challenger first computes $(PK_{\mathcal{T}}, SK_{\mathcal{T}}, m_B, m_C) \leftarrow \mathsf{Setup}(k)$ and $(pk, sk) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$. $\mathcal{A}$ is given as input $(PK_{\mathcal{T}}, pk, k)$ and forwards $(PK_{\mathcal{T}}, k)$ to $\mathcal{A}'$. $\mathcal{A}'$ outputs n_V.

Now $\mathcal{A}'$ may make some oracle queries. $\mathcal{A}$ will maintain a list H of (i, pk'_i, sk'_i) tuples. $\mathcal{A}'$'s first oracle, C, needs to return secret keys associated with the pk_i. When $\mathcal{A}$ receives a query $C(i)$, $\mathcal{A}$ checks if $(i, pk'_i, sk'_i) \in H$. If so, $\mathcal{A}$ returns sk'_i. Otherwise, $\mathcal{A}$ computes $(pk'_i, sk'_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$, adds (i, pk'_i, sk'_i) to H, and returns sk'_i. Again by the IND-CPA property of the encryption scheme, $\mathcal{A}'$ cannot tell that sk'_i does not actually correspond to pk_i. $\mathcal{A}'$'s second oracle, R can be queried on inputs (i, β, n_C), on which it returns $\mathsf{Vote}(sk_i, PK_{\mathcal{T}}, n_C, \beta, k)$. If $\mathcal{A}$ receives the query $R(i, \beta, n_C)$, it checks if $(i, pk'_i, sk'_i) \in H$. If so, $\mathcal{A}$ computes $b = \mathsf{Vote}(sk'_i, PK_{\mathcal{T}}, Q, n_C, \beta, k)$ and returns b. Otherwise, $\mathcal{A}$ computes $(pk'_i, sk'_i) \leftarrow \mathsf{Register}(PK_{\mathcal{T}}, k)$, adds (i, pk'_i, sk'_i) to H, then computes $b = \mathsf{Vote}(sk'_i, PK_{\mathcal{T}}, n_C, \beta, k)$ and returns b. Again by the IND-CPA property of the encryption scheme, $\mathcal{A}'$ cannot tell that the ballots b he receives were computed with a secret key that does not correspond to pk_i. Finally, $\mathcal{A}'$ outputs (n_C, β, i, b), and $\mathcal{A}$ outputs (n_C, β, b). Clearly, $\mathcal{A}$ has the same success probability as $\mathcal{A}'$, so $\mathcal{A}$ wins Exp-EV-1-Int with non-negligible probability. □

3) Eligibility Verifiability:

Proposition 33. *Suppose Γ is a homomorphic asymmetric encryption scheme, Σ_1, Σ_2, Σ_3, Σ_4, Σ_5 and Σ_6, are sigma protocols and $\mathcal{H}$ is a hash function such that the conditions of Figure 3 are satisfied. Further suppose that Γ satisfies IND-CPA. We have $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$ satisfies eligibility verifiability.*

Proof. Let $\Pi_{JCJ} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be defined as above. Let $\mathcal{A}'$ be an adversary that wins the Exp-EV-1-Int game. We will construct the adversary $\mathcal{A}$ that wins the IND-1-CPA game with non-negligible advantage. The challenger first generates $(PK, SK_{\mathcal{T}}, \mathfrak{m}) \leftarrow \mathsf{Gen}'(k)$, where $PK = (PK_{\mathcal{T}}, k)$ and $PK_{\mathcal{T}} = (pk_T, \mathfrak{m}, \rho)$, and gives $(PK_{\mathcal{T}}, k)$ to $\mathcal{A}$ as input. $\mathcal{A}$ runs $\mathsf{Register}(PK_{\mathcal{T}}, k)$ twice to get $(pk_0, sk_0), (pk_1, sk_1)$ and sets $m_0 = (1, sk_0), m_1 = (1, sk_1)$. $\mathcal{A}$ then outputs (m_0, m_1). The challenger picks a bit b at random and gives $c = \mathsf{Enc}'(PK, m_b)$ to $\mathcal{A}$. We have $c = (c_1, c_2, \tau)$, where $c_2 = \mathsf{Enc}(pk_T, sk_b)$. Now $\mathcal{A}$ begins to interact with $\mathcal{A}'$ by giving $(PK_{\mathcal{T}}, c_2, k)$ to $\mathcal{A}'$.

At this point $\mathcal{A}'$ may call its oracle R. If $\mathcal{A}$ receives a query $R(\beta, n_C)$, it will construct $x \leftarrow \mathsf{Vote}(sk_0, PK_{\mathcal{T}}, n_C, \beta, k)$ and return x. We have that $x = (\mathsf{Enc}(pk_T, \beta), \mathsf{Enc}(pk_T, sk_0), \sigma, \tau)$, where σ and τ are proofs of plaintext knowledge in a subspace and conjunctive plaintext knowledge, respectively. By the IND-CPA property of Enc, $\mathcal{A}'$ can't distinguish between encryptions of sk_0 and sk_1. Therefore we can construct x using sk_0 even if the secret key corresponding to c_2 is actually sk_1.

$\mathcal{A}'$ will then output (n_C, β, b^*), where $b^* = (c_1^*, c_2^*, \sigma^*, \tau^*)$. $\mathcal{A}'$ wins with probability $\frac{1}{p(k)}$ for some polynomial function p, so with probability $\frac{1}{p(k)}$ we have that $c_1 = \mathsf{Enc}(pk_T, \beta; r_1), c_2 = \mathsf{Enc}(pk_T, sk_b; r_2)$, $\sigma = \mathsf{ProveCiph}((pk_T, c_1, \{1, \dots, n_C\}), (\beta, r_1), k)$, and $\tau^* = \mathsf{ProveBind}((pk_T, c_1, c_2), (\beta, r_1, sk_b, r_2), k)$. In order to ensure that we get a ballot of this form from $\mathcal{A}'$ with high enough probability, $\mathcal{A}$ repeats the above interaction with $\mathcal{A}'$ $p(k)$ times to obtain $(n_C^1, \beta^1, b_1^*), \dots, (n_C^{p(k)}, \beta^{p(k)}, b_{p(k)}^*)$. $\mathcal{A}$ outputs $(b_1^*[1], b_1^*[2], b_1^*[4]), \dots, (b_{p(k)}^*[1], b_{p(k)}^*[2], b_{p(k)}^*[4])$, and receives $\mathsf{Dec}'(SK_{\mathcal{T}}, (b_1^*[1], b_1^*[2], b_1^*[4])), \dots, \mathsf{Dec}'(SK_{\mathcal{T}}, (b_{p(k)}^*, b_{p(k)}^*[2], b_{p(k)}^*[4]))$. If there exist i, j such that $\mathsf{Dec}'(SK_{\mathcal{T}}, (b_i^*[1], b_i^*[2], b_i^*[4])) = (\beta^i, sk_0)$ and $\mathsf{Dec}'(SK_{\mathcal{T}}, (b_j^*[1], b_j^*[2], b_j^*[4])) = (\beta^j, sk_1)$, or there exists no i such that $\mathsf{Dec}'(SK_{\mathcal{T}}, (b_i^*[1], b_i^*[2], b_i^*[4])) = (\beta^*, sk_0)$ or (β^*, sk_1), then $\mathcal{A}$ outputs a random bit. Otherwise, if there exists i such that $\mathsf{Dec}'(SK_{\mathcal{T}}, (b_i^*[1], b_i^*[2], b_i^*[4])) = (\beta^*, sk_0)$, then $\mathcal{A}$ outputs 0. Likewise, if there exists i such that $\mathsf{Dec}'(SK_{\mathcal{T}}, (b_i^*[1], b_i^*[2], b_i^*[4])) = (\beta^*, sk_1)$, then $\mathcal{A}$ outputs 1.

We now argue that $\mathcal{A}$ can determine the correct bit b with non-negligible advantage.

There are three possible events that can occur in a run of $\mathcal{A}$. The first possibility is that $\mathcal{A}'$ fails on each of its $p(k)$ runs so that $\mathcal{A}$ has to guess. This occurs with probability $(1-\frac{1}{p(k)})^{p(k)}$. The second event is that $\mathcal{A}'$ does succeed in one of its runs, but on a different run it outputs

$$\begin{aligned} b = (&\mathsf{Enc}(PK_{\mathcal{T}}, \beta; r_1),\\ &\mathsf{Enc}(PK_{\mathcal{T}}, sk_{(1-b)}; r_2),\\ &\mathsf{ProveCiph}((PK_{\mathcal{T}}, c_1, \{1, \dots, n_C\}), (\beta, r_1), k),\\ &\mathsf{ProveBind}((PK_{\mathcal{T}}, c_1, c_2), (\beta, r_1, sk_{(1-b)}, r_2), k)). \end{aligned}$$

However, because sk_0 and sk_1 are chosen randomly, the probability of this occurring is negligible. Finally, the third possibility is that $\mathcal{A}'$ succeeds in at least one of its runs. This occurs with probability $\sum_{i=0}^{p(k)-1}(1-\frac{1}{p(k)})^i(\frac{1}{p(k)})$. In the first two events, $\mathcal{A}$ guesses and wins with probability $\frac{1}{2}$, and in the third event $\mathcal{A}$ wins with probability 1. Therefore, the total probability that $\mathcal{A}$ wins is $(\sum_{i=0}^{p(k)-1}(1-\frac{1}{p(k)})^i(\frac{1}{p(k)})) + \frac{1}{2}(1-\frac{1}{p(k)})^{p(k)} + \frac{1}{2}\mu(k)$, for some negligible function μ.

We have that this equation is equal to:

$$\begin{aligned} &= \frac{1}{p(k)}\sum_{i=0}^{p(k)-1}(1-\frac{1}{p(k)})^i + \frac{1}{2}(1-\frac{1}{p(k)})^{p(k)}\\ &\quad +\frac{1}{2}\mu(k)\\ &= \frac{1}{p(k)}(p(k) - (1-\frac{1}{p(k)})^{p(k)}p(k)) + \frac{1}{2}(1-\frac{1}{p(k)})^{p(k)}\\ &\quad +\frac{1}{2}\mu(k)\\ &= 1-(1-\frac{1}{p(k)})^{p(k)} + \frac{1}{2}(1-\frac{1}{p(k)})^{p(k)} + \frac{1}{2}\mu(k)\\ &= 1-\frac{1}{2}(1-\frac{1}{p(k)})^{p(k)} + \frac{1}{2}\mu(k) \end{aligned}$$

In order to determine the advantage of this adversary, we subtract $\frac{1}{2}$ from this:

$$1 - \frac{1}{2}(1 - \frac{1}{p(k)})^{p(k)} + \frac{1}{2}\mu(k) - \frac{1}{2}$$
$$= \frac{1}{2} - \frac{1}{2}(1 - \frac{1}{p(k)})^{p(k)} + \frac{1}{2}\mu(k)$$

As k gets large, $(1 - \frac{1}{p(k)})^{p(k)}$ converges to $\frac{1}{e}$ and $\mu(k)$ goes to 0, so the entire equation converges to $\frac{1}{2} - \frac{1}{2e}$. This is non-negligible.

Combining this reduction with Lemma 32, we have that if Π_{JCJ} satisfies IND-1-CPA, then JCJ satisfies Exp-EV-Int-Weak. □

D. Election verifiability

By Propositions 29, 30, & 33, election schemes constructed from generalized JCJ satisfy election verifiability with internal authentication:

Corollary 34. *Suppose* Γ, Σ_1, Σ_2, Σ_3, Σ_4, Σ_5, Σ_6 *and* $\mathcal{H}$ *satisfy the preconditions of Figure 3. Further suppose that* Γ *satisfies IND-CPA and is collision-free,* Σ_1 *and* Σ_6 *satisfy special soundness and special honest verifier zero-knowledge, and* $\mathcal{H}$ *is a random oracle. We have* $\mathsf{JCJ}(\Gamma, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_6, \mathcal{H})$ *satisfies election verifiability with internal authentication.*

E. Proof: Theorem 8

Proof of Theorem 8. We know that Γ satisfies IND-CPA, and by Corollary 28 Γ is also collision-free. Therefore the proof follows from Corollary 34, subject to the applicability of Theorem 12 to the mixnet and sigma protocol used by JCJ to prove correct key construction. □

APPENDIX J
JUELS ET AL. DEFINITIONS

Juels et al. [83, §2] define an election scheme as a tuple of (Register, Vote, Tally, Verify) PPT algorithms:

- **Register**, denoted $(pk, sk) \leftarrow \mathsf{Register}(SK_{\mathcal{R}}, i, k_1)$, is executed by the registrars. Register takes as input the private key $SK_{\mathcal{R}}$ of the registrars, a voter's identity i, and security parameter k_1. It outputs a credential pair (pk, sk).
- **Vote**, denoted $b \leftarrow \mathsf{Vote}(sk, PK_{\mathcal{T}}, n_C, \beta, k_2)$, is executed by voters. Vote takes as input a voter's private credential sk, the public key $PK_{\mathcal{T}}$ of the tallier, the number of candidates n_C, the voter's choice β, and security parameter k_2. It outputs a ballot b.
- **Tally**, denoted $(\mathbf{X}, P) \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$, is executed by the tallier. Tally takes as input the private key $SK_{\mathcal{T}}$ of the tallier, the bulletin board BB, the number of candidates n_C, the set containing voters' public credentials, and security parameter k_3. It outputs the tally $\mathbf{X}$ and a proof P that the tally is correct.
- **Verify**, denoted $v \leftarrow \mathsf{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P)$, can be executed by anyone to audit the election. Verify takes as input the public key $PK_{\mathcal{R}}$ of the registrars, the public key $PK_{\mathcal{T}}$ of the tallier, the bulletin board BB, the number of candidates n_C, and a candidate proof P of correct tallying. It outputs a bit v, which is 1 if the tally successfully verifies and 0 on failure.

The above definition fixes an apparent oversight in JCJ's presentation: we supply the registrars' public key as input to the verification algorithm, because that key would be required by Verify to check the signature on the electoral roll.

Juels et al. [83, §3] formalize *correctness* and *verifiability* to capture their notion of election verifiability. We rename those to *JCJ-correctness* and *JCJ-verifiability* to avoid ambiguity. For readability, the definitions we give below contain subtle differences from the original presentation. For example, we sometimes use for loops instead of pattern matching.

JCJ-correctness asserts that an adversary cannot modify or eliminate votes of honest voters, and stipulates that at most one ballot is tallied per voter. Intuitively, the security definition challenges the adversary to ensure that verification succeeds and the tally[49] does not include some honest votes or contains too many votes. The definition of JCJ-correctness fixes apparent errors in the original presentation: the adversary is given the credentials for corrupt voters and distinct security parameters are supplied to the Register and Vote algorithms. An implicit assumption is also omitted: $\{\beta_i\}_{i\in\mathcal{V}\setminus\mathcal{V}'}$ is a multiset of valid votes, that is, for all $\beta \in \{\beta_i\}_{i\in\mathcal{V}\setminus\mathcal{V}'}$ we have $1 \leq \beta \leq n_C$. Without this assumption the security definition cannot be satisfied by many election schemes, including the election scheme by Juels et al.

Definition 31 (JCJ-correctness)**.** *An election scheme* $\Pi = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* JCJ-correctness *if for all PPT adversary* $\mathcal{A}$*, there exists a negligible function* μ*, such that for all positive integers* n_C *and* n_V*, and security parameters* k_1*,* k_2*, and* k_3*, we have* $\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \leq \mu(k_1, k_2, k_3)$*, where* Exp-JCJ-Cor *is defined as follows:*[50]

$\mathsf{Exp\text{-}JCJ\text{-}Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3) =$

```
V ← {1, ..., n_V};
for i ∈ V do (pk_i, sk_i) ← Register(SK_R, i, k_1)
V' ← A({pk_i}_{i=1}^{n_V});
for i ∈ V \ V' do β_i ← A();
BB ← {Vote(sk_i, PK_T, n_C, β_i, k_2)}_{i∈V\V'};
(X, P) ← Tally(SK_T, BB, n_C, {pk_i}_{i=1}^{n_V}, k_3);
BB ← BB ∪ A(BB, {(pk_i, sk_i)}_{i∈V∩V'});
(X', P') ← Tally(SK_T, BB, n_C, {pk_i}_{i=1}^{n_V}, k_3);
if Verify(PK_R, PK_T, BB, n_C, X', P') = 1
  ∧ ({β_i}_{i∈V\V'} ⊈ ⟨X'⟩ ∨ |⟨X'⟩| − |⟨X⟩| > |V'|) then
    return 1
else
    return 0
```

The JCJ-correctness definition implicitly assumes that the tally and associated proof are honestly computed using

49. Juels et al. translate tallies $\mathbf{X}$ into a multisets $\langle\mathbf{X}\rangle$ representing the tally as follows: $\langle\mathbf{X}\rangle = \bigcup_{1\leq j\leq|\mathbf{X}|}\{\underbrace{j, \ldots, j}_{\mathbf{X}[j] \text{ times}}\}$.

50. We write $\mu(k_1, k_2, k_3)$ for the smallest value in $\{\mu(k_1), \mu(k_2), \mu(k_3)\}$ (cf. [83, pp45]).

the Tally algorithm. By comparison, the definition of JCJ-verifiability (Definition 32) does not use this assumption, hence, JCJ-verifiability is intended to assert that voters and auditors can check whether votes have been recorded and tallied correctly. Intuitively, the adversary is assumed to control the tallier and voters, and the security definition challenges the adversary to concoct an election (that is, the adversary generates a bulletin board BB, a tally $\mathbf{X}$, and a proof of tallying P) such that verification succeeds and tally $\mathbf{X}$ differs tally $\mathbf{X}'$ derived from honestly tallying the bulletin board BB. It follows that there is at most one verifiable tally that can be derived.

Definition 32 (JCJ-verifiability)**.** *An election scheme* $\Pi = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ *satisfies* JCJ-verifiability *if for all PPT adversary $\mathcal{A}$, there exists a negligible function μ, such that for all positive integers n_C and n_V, and security parameters k_1 and k_3, we have* $\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Ver}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \leq \mu(k_1, k_2, k_3)$, *where* Exp-JCJ-Ver *is defined as follows:*

$\mathsf{Exp\text{-}JCJ\text{-}Ver}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3) =$

for $1 \leq i \leq n_V$ **do** $(pk_i, sk_i) \leftarrow \mathsf{Register}(SK_{\mathcal{R}}, i, k_1)$
$(BB, \mathbf{X}, P) \leftarrow \mathcal{A}(SK_{\mathcal{T}}, \{(pk_i, sk_i)\}_{i=1}^{n_V})$;
$(\mathbf{X}', P') \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$;
if $\mathsf{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) = 1 \wedge \mathbf{X} \neq \mathbf{X}'$ **then**
  return 1
else
  return 0

Appendix K
Proofs: Juels et al. Admit Attacks

This appendix contains proofs demonstrating that the definition of election verifiability by Juels et al. [83] admits collusion and biasing attacks (§VII). We have reported these findings to the original authors.[51,52]

A. Proof: Proposition 9

Suppose $\Pi = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ is an election scheme satisfying JCJ-correctness and JCJ-verifiability. Further suppose $\mathsf{Stuff}(\Pi, \beta, \kappa) = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}_S, \mathsf{Verify}_S)$, for some integers $\beta, \kappa \in \mathbb{N}$. We prove that $\mathsf{Stuff}(\Pi, \beta, \kappa)$ satisfies JCJ-correctness and JCJ-verifiability.

We show that $\mathsf{Stuff}(\Pi, \beta, \kappa)$ satisfies JCJ-correctness by contradiction. Suppose $\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Cor}(\mathsf{Stuff}(\Pi, \beta, \kappa), \mathcal{A}, n_C, n_V, k_1, k_2, k_3))$ is non-negligible for some k_1, k_2, k_3, n_C, n_V, and $\mathcal{A}$. Hence, there exists an execution of the experiment

$$\mathsf{Exp\text{-}JCJ\text{-}Cor}(\mathsf{Stuff}(\Pi, \beta, \kappa), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$$

that satisfies

$$\mathsf{Verify}_S(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}', P') = 1 \\ \wedge \left(\{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'} \not\subseteq \langle \mathbf{X}' \rangle \vee |\langle \mathbf{X}' \rangle| - |\langle \mathbf{X} \rangle| > |\mathcal{V}'|\right)$$

with non-negligible probability, where $\{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'}$ is the set of honest votes, $(\mathbf{X}, P)$ is the tally of honest votes, $(\mathbf{X}', P')$ is the tally of all votes, $\mathcal{V}'$ is a set of corrupt voter identities, and BB is the bulletin board. Further suppose BB_0 is the bulletin board BB before adding stuffed ballots. By definition of Tally_S, there exist computations

$$(\mathbf{Y}, Q) \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB_0, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$$

and

$$(\mathbf{Y}', Q') \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$$

such that $\mathbf{X} = \mathtt{Add}(\mathbf{Y}, \beta, \kappa)$, $\mathbf{X}' = \mathtt{Add}(\mathbf{Y}', \beta, \kappa)$, and $P' = Q'$. Since $\kappa \in \mathbb{N}$, we have $\langle \mathbf{Y}' \rangle \subseteq \langle \mathbf{X}' \rangle$. Moreover, $|\langle \mathbf{X} \rangle| = |\langle \mathbf{Y} \rangle| + \kappa$ and $|\langle \mathbf{X}' \rangle| = |\langle \mathbf{Y}' \rangle| + \kappa$, hence,

$$|\langle \mathbf{Y}' \rangle| - |\langle \mathbf{Y} \rangle| = |\langle \mathbf{X}' \rangle| - |\langle \mathbf{X} \rangle|.$$

By definition of Verify_S and since $\mathbf{Y}' = \mathtt{Sub}(\mathbf{X}', \beta, \kappa)$, there exists a computation

$$v \leftarrow \mathsf{Verify}_0(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{Y}', Q')$$

such that $v = 1$. It follows that

$$\mathsf{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{Y}', Q') = 1 \\ \wedge \left(\{\beta_i\}_{i \in \mathcal{V} \setminus \mathcal{V}'} \not\subseteq \langle \mathbf{Y}' \rangle \vee |\langle \mathbf{Y}' \rangle| - |\langle \mathbf{Y} \rangle| > |\mathcal{V}'|\right)$$

with non-negligible probability and, furthermore, we have $\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3))$ is non-negligible, thereby deriving a contradiction.

We show that $\mathsf{Stuff}(\Pi, \beta, \kappa)$ satisfies JCJ-verifiability by contradiction. Suppose $\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Ver}(\mathsf{Stuff}(\Pi, \beta, \kappa), \mathcal{A}, n_C, n_V, k_1, k_2, k_3))$ is non-negligible for some k_1, k_3, n_C, n_V, and $\mathcal{A}$. Hence, there exists an execution of the experiment $\mathsf{Exp\text{-}JCJ\text{-}Ver}(\mathsf{Stuff}(\Pi, \beta, \kappa), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$ which satisfies

$$\mathsf{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) = 1 \wedge \mathbf{X} \neq \mathbf{X}'$$

with non-negligible probability, where $(BB, \mathbf{X}, P)$ is an election concocted by the adversary and $(\mathbf{X}', P')$ is produced by tallying BB. By definition of Tally_S, there exists a computation

$$(\mathbf{Y}', Q') \leftarrow \mathsf{Tally}(SK_{\mathcal{T}}, BB, n_C, \{pk_i\}_{i=1}^{n_V}, k_3)$$

such that $\mathbf{X}' = \mathtt{Add}(\mathbf{Y}', \beta, \kappa)$ and $P' = Q'$. By definition of Verify_S, there exists a computation

$$v \leftarrow \mathsf{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathtt{Sub}(\mathbf{X}, \beta, \kappa), P)$$

such that $v = 1$. Let the adversary $\mathcal{B}$ be defined as follows: given input K and S, the adversary $\mathcal{B}$ computes

$$(BB, \mathbf{X}, P) \leftarrow \mathcal{A}(K, S)$$

and outputs $(BB, \mathtt{Sub}(\mathbf{X}, \beta, \kappa), P)$. We have an execution of the experiment $\mathsf{Exp\text{-}JCJ\text{-}Ver}(\mathsf{Stuff}(\Pi, \beta, \kappa), \mathcal{B}, n_C, n_V, k_1,$

51. Dario Catalano, personal communication, Paris, France, 10 October 2013.

52. Markus Jakobsson, personal communication, New Orleans, USA, 27 June 2013.

$k_2, k_3)$ that concocts the election $(BB, \texttt{Sub}(\mathbf{X}, \beta, \kappa), P)$ and tallying BB produces $(\mathbf{Y}', Q')$ such that

$$\mathsf{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \texttt{Sub}(\mathbf{X}, \beta, \kappa), P) = 1$$

with non-negligible probability. Moreover, since $\mathbf{X} \neq \mathbf{X}'$ and $\mathbf{Y}' = \texttt{Sub}(\mathbf{X}', \beta, \kappa)$, we have $\texttt{Sub}(\mathbf{X}, \beta, \kappa) \neq \mathbf{Y}'$ with non-negligible probability. It follows immediately that $\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Cor}(\Pi, \mathcal{B}, n_C, n_V, k_1, k_2, k_3))$ is non-negligible, thus deriving a contradiction and concluding our proof. □

B. Proof: Proposition 10

We define key leakage before proving Proposition 10.

Definition 33 (Key leakage)**.** *An election scheme* $\Pi = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ does not leak the tallier's private key *if for all positive integers* n_C *and* n_V*, security parameters* k_1 *and* k_3*, and PPT adversary* $\mathcal{A}$*, we have* $\mathsf{Succ}(\mathsf{Exp\text{-}leak}(\Pi, \mathcal{A}, k_1, k_3, n_C, n_V))$ *is negligible, where* $\mathsf{Exp\text{-}leak}(\cdot)$ *is defined as follows:*

$\mathsf{Exp\text{-}leak}(\Pi, \mathcal{A}, k_1, k_3, n_C, n_V) =$

```
for 1 ≤ i ≤ n_V do (pk_i, sk_i) ← Register(SK_R, i, k_1);
BB ← A(sk_1, ..., sk_{n_V});
(X, P) ← Tally(SK_T, BB, n_C, {pk_i}_{i=1}^{n_V}, k_3);
if A(X, P) = SK_T then
  | return 1
else
  | return 0
```

Proof of Proposition 10. Suppose $\Pi = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ is an election scheme satisfying JCJ-correctness and JCJ-verifiability that does not leak the tallier's private key, and $\mathsf{Backdoor}(\Pi) = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}_B, \mathsf{Verify}_B)$. Since Π does not leak the tallier's private key, the experiment $\mathsf{Exp\text{-}JCJ\text{-}Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$ produces a bulletin board BB such that $SK_{\mathcal{T}} \in BB$ with negligible probability, for all k_1, k_2, k_3, n_C, n_V and $\mathcal{A}$. It follows that an execution of the experiment $\mathsf{Exp\text{-}JCJ\text{-}Cor}(\mathsf{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$ is an execution of the experiment $\mathsf{Exp\text{-}JCJ\text{-}Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$ with non-negligible probability and, hence, $\mathsf{Backdoor}(\Pi)$ satisfies JCJ-correctness.

We show that $\mathsf{Backdoor}(\Pi)$ satisfies JCJ-verifiability by contradiction. Suppose $\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Ver}(\mathsf{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3))$ is non-negligible for some k_1, k_3, n_C, n_V, and $\mathcal{A}$. Hence, there exists an execution of the experiment $\mathsf{Exp\text{-}JCJ\text{-}Ver}(\mathsf{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$ which satisfies

$$\mathsf{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) = 1 \wedge \mathbf{X} \neq \mathbf{X}'$$

with non-negligible probability, where $(BB, \mathbf{X}, P)$ is an election concocted by the adversary and $(\mathbf{X}', P')$ is produced by tallying BB. If there is one and only one $\mathbf{Y}$ such that $(SK_{\mathcal{T}}, \mathbf{Y}) \in BB$, then $\mathbf{X}' = \mathbf{Y}$ by definition of Tally and $\mathbf{X} = \mathbf{Y}$ by definition of Verify, otherwise, the execution of the experiment $\mathsf{Exp\text{-}JCJ\text{-}Cor}(\mathsf{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$ is an execution of the experiment $\mathsf{Exp\text{-}JCJ\text{-}Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)$ and, hence,

$$\begin{aligned}&\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Ver}(\mathsf{Backdoor}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \\ &\quad = \mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Ver}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3)).\end{aligned}$$

In both cases we derive a contradiction, thereby concluding our proof. □

C. Proof sketch: Proposition 11

Suppose $\Pi = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify})$ is an election scheme satisfying JCJ-correctness and JCJ-verifiability. Further suppose $\mathsf{Bias}(\Pi, Z) = (\mathsf{Register}, \mathsf{Vote}, \mathsf{Tally}, \mathsf{Verify}_R)$, for some set of vectors Z. By definition of Verify_R, we have

$$\mathsf{Verify}_R(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P) = 1$$

implies the existence of a computation

$$v \leftarrow \mathsf{Verify}(PK_{\mathcal{R}}, PK_{\mathcal{T}}, BB, n_C, \mathbf{X}, P)$$

such that $v = 1$ with non-negligible probability, for all $PK_{\mathcal{T}}$, BB, n_C, $\mathbf{X}$, and P. It follows that

$$\begin{aligned}&\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Cor}(\mathsf{Bias}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \\ &\quad \leq \mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Cor}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3))\end{aligned}$$

and

$$\begin{aligned}&\mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Ver}(\mathsf{Bias}(\Pi), \mathcal{A}, n_C, n_V, k_1, k_2, k_3)) \\ &\quad \leq \mathsf{Succ}(\mathsf{Exp\text{-}JCJ\text{-}Ver}(\Pi, \mathcal{A}, n_C, n_V, k_1, k_2, k_3))\end{aligned}$$

for all k_1, k_2, k_3, n_C, n_V, and $\mathcal{A}$. Hence, $\mathsf{Bias}(\Pi, Z)$ satisfies JCJ-correctness and JCJ-verifiability. □

References

[1] Ben Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2006.

[2] Ben Adida. Helios: Web-based Open-Audit Voting. In *USENIX Security'08: 17th USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[3] Ben Adida. Helios deployed at Princeton. http://heliosvoting.wordpress.com/2009/10/13/helios-deployed-at-princeton/ (accessed 7 May 2014), 2009.

[4] Ben Adida. Helios v4 Verification Specs. Helios documentation, http://documentation.heliosvoting.org/verification-specs/helios-v4 (accessed 2 May 2014), 2014. A snapshot of the specification on 8 Oct 2013 is available from https://web.archive.org/web/20131018033747/http://documentation.heliosvoting.org/verification-specs/helios-v4.

[5] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In *EVT/WOTE'09: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Association, 2009.

[6] Ben Adida and C. Andrew Neff. Ballot casting assurance. In *EVT'06: Electronic Voting Technology Workshop*, Berkeley, CA, USA, 2006. USENIX Association.

[7] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT'02: 21st International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2332 of *LNCS*, pages 83–107. Springer, 2002.

[8] Myrto Arapinis, Sergiu Bursuc, and Mark Ryan. Reduction of Equational Theories for Verification of Trace Equivalence: Re-encryption, Associativity and Commutativity. In *POST'12: First Conference on Principles of Security and Trust*, volume 7215 of *LNCS*, pages 169–188. Springer, 2012.

[9] Michael Backes, Cătălin Hriţcu, and Matteo Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In *CSF'08: 21st Computer Security Foundations Symposium*, pages 195–209. IEEE Computer Society, 2008.

[10] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO'98: 18th International Cryptology Conference*, volume 1462 of *LNCS*, pages 26–45. Springer, 1998.

[11] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93: 1st ACM Conference on Computer and Communications Security*, pages 62–73, New York, NY, USA, 1993. ACM.

[12] Mihir Bellare and Amit Sahai. Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *CRYPTO'99: 19th International Cryptology Conference*, volume 1666 of *LNCS*, pages 519–536. Springer, 1999.

[13] Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. Cryptology ePrint Archive, Report 2006/228, 2006.

[14] Josh Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Department of Computer Science, Yale University, 1996.

[15] Josh Benaloh. Simple Verifiable Elections. In *EVT'06: Electronic Voting Technology Workshop*. USENIX Association, 2006.

[16] Josh Benaloh. Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In *EVT'07: Electronic Voting Technology Workshop*. USENIX Association, 2007.

[17] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC '94: Twenty-sixth Annual ACM Symposium on Theory of Computing*, pages 544–553, New York, NY, USA, 1994. ACM Press.

[18] Josh Benaloh, Serge Vaudenay, and Jean-Jacques Quisquater. Final Report of IACR Electronic Voting Committee. International Association for Cryptologic Research. http://www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html (accessed 7 May 2014), Sept 2010.

[19] Josh Benaloh and Moti Yung. Distributing the Power of a Government to Enhance the Privacy of Voters. In *PODC'86: 5th Principles of Distributed Computing Symposium*, pages 52–62. ACM Press, 1986.

[20] David Bernhard. *Zero-Knowledge Proofs in Theory and Practice.* PhD thesis, Department of Computer Science, University of Bristol, 2014.

[21] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *ASIACRYPT'12: 18th International Conference on the Theory and Application of Cryptology and Information Security*, volume 7658 of *LNCS*, pages 626–643. Springer, 2012.

[22] David Bernhard, Olivier Pereira, and Bogdan Warinschi. On Necessary and Sufficient Conditions for Private Ballot Submission. Cryptology ePrint Archive, Report 2012/236 (version 20120430:154117b), 2012.

[23] Bruno Blanchet, Ben Smyth, and Vincent Cheval. *ProVerif 1.91: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2015. Originally appeared as Bruno Blanchet and Ben Smyth (2011) ProVerif 1.85: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial.

[24] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational diffie-hellman. In *PKC'06: 9th International Workshop on Practice and Theory in Public Key Cryptography*, pages 229–240. Springer, 2006.

[25] Debra Bowen. Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. California Secretary of State, press release DB07:042 http://www.sos.ca.gov/voting-systems/oversight/ttbr/db07-042-ttbr-system-decisions-release.pdf (accessed 7 May 2014), August 2007.

[26] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *FOCS'01: 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, October 2001.

[27] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In *CSF'14: 27th Computer Security Foundations Symposium*. IEEE Computer Society, 2014. To appear.

[28] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[29] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, 2004.

[30] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity II: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVT'08: Electronic Voting Technology Workshop*, pages 14:1–14:13. USENIX Association, 2008.

[31] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf, and René Peralta. Demonstrating Possession of a Discrete Logarithm Without Revealing It. In *CRYPTO'86: 6th International Cryptology Conference*, volume 263 of *LNCS*, pages 200–212. Springer, 1987.

[32] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. In *CRYPTO'92: 12th International Cryptology Conference*, volume 740 of *LNCS*, pages 89–105. Springer, 1993.

[33] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A Practical Voter-Verifiable Election Scheme. In *ESORICS'05: 10th European Symposium On Research In Computer Security*, volume 3679 of *LNCS*, pages 118–139. Springer, 2005.

[34] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On Some Incompatible Properties of Voting Schemes. In *WOTE'06: Workshop on Trustworthy Elections*, 2006.

[35] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On Some Incompatible Properties of Voting Schemes. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 191–199. Springer, 2010.

[36] Michael Clarkson, Brian Hay, Meador Inge, abhi shelat, David Wagner, and Alec Yasinsac. Software review and security analysis of Scytl remote voting software. Report commissioned by Florida Division of Elections. Available from http://election.dos.state.fl.us/voting-systems/pdf/FinalReportSept19.pdf. Filed September 19, 2008.

[37] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. In *S&P'08: 29th Security and Privacy Symposium*, pages 354–368. IEEE Computer Society, 2008.

[38] Josh Daniel Cohen and Michael J. Fischer. A Robust and Verifiable Cryptographically Secure Election Scheme. In *FOCS'85: 26th IEEE Symposium on Foundations of Computer Science*, pages 372–382. IEEE Computer Society, 1985.

[39] Véronique Cortier, Fabienne Eigner, Steve Kremer, Matteo Maffei, and Cyrille Wiedling. Type-Based Verification of Electronic Voting Protocols. In *POST'15: 4th Conference on Principles of Security and Trust*, LNCS, pages 303–323. Springer, 2015.

[40] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. Distributed elgamal à la pedersen - application to helios. In *WPES'13: Workshop on Privacy in the Electronic Society*, pages 131–142. ACM Press, 2013.

[41] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election Verifiability for Helios under Weaker Trust Assumptions. In *ESORICS'14: 19th European Symposium on Research in Computer Security*, volume 8713 of *LNCS*, pages 327–344. Springer, 2014.

[42] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election Verifiability for Helios under Weaker Trust Assumptions. Technical Report RR-8555, INRIA, 2014.

[43] Veronique Cortier, David Galindo, Ralf Kuesters, Johannes Mueller, and Tomasz Truderung. Verifiability Notions for E-Voting Protocols. Cryptology ePrint Archive, Report 2016/287 (version 20160317:161048), 2016.

[44] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 297–311. IEEE Computer Society, 2011.

[45] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013.

[46] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *CRYPTO'94: 14th International Cryptology Conference*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.

[47] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-Autority Secret-Ballot Elections with Linear Work. In *EUROCRYPT'96: 15th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1070 of *LNCS*, pages 72–83. Springer, 1996.

[48] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *EUROCRYPT'97: 16th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.

[49] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO'98: 18th International Cryptology Conference*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.

[50] Chris Culnane and Steve A. Schneider. A Peered Bulletin Board for Robust Use in Verifiable Voting Systems. In *CSF'14: 27th Computer Security Foundations Symposium*, pages 169–183. IEEE Computer Society, 2014.

[51] Ivan Damgård. On Σ-protocols, 2010. Available from http://www.daimi.au.dk/~ivan/Sigma.pdf.

[52] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A Generalization of Paillier's Public-Key System with Applications to Electronic Voting. *International Journal of Information Security*, 9(6):371–385, 2010.

[53] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[54] Stéphanie Delaune, Steve Kremer, Mark D. Ryan, and Graham Steel. Formal analysis of protocols based on TPM state registers. In *CSF'11: 24th Computer Security Foundations Symposium*, pages 66–80. IEEE Computer Society, 2011.

[55] Jannik Dreier, Rosario Giustolisi, Ali Kassem, Pascal Lafourcade, and Gabriele Lenzini. On the Verifiability of (Electronic) Exams. Technical Report TR-2014-2, Verimag, 2014.

[56] Jannik Dreier, Hugo Jonker, and Pascal Lafourcade. Defining verifiability in e-auction protocols. In *ASIACCS'13: 8th ACM Symposium on Information, Computer and Communications Security*, pages 547–552. ACM Press, 2013.

[57] Jannik Dreier, Hugo Jonker, and Pascal Lafourcade. A framework for analyzing verifiability in traditional and electronic exams. In *ISPEC'15: 11th International Conference on Information Security Practice and Experience*, volume 9065 of *LNCS*, pages 514–529. Springer, 2015.

[58] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[59] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86: 6th International Cryptology Conference*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[60] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques*, volume 718 of *LNCS*, pages 244–251. Springer, 1992.

[61] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *CRYPTO'01: 21st International Cryptology Conference*, volume 2139 of *LNCS*. Springer, 2001.

[62] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT'99: 18th International Conference on the Theory and Applications of Cryptographic Techniques*, pages 295–310. Springer, 1999.

[63] Use of voting computers in 2005 Bundestag election unconstitutional, March 2009. Press release 19/2009 http://www.bundesverfassungsgericht.de/en/press/bvg09-019en.html (accessed 7 May 2014).

[64] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, Cambridge, UK, 2001.

[65] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information. In *STOC'82: 14th Annual ACM Symposium on Theory of Computing*, pages 365–377. ACM Press, 1982.

[66] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[67] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[68] Jens Groth. Evaluating security of voting schemes in the universal composability framework. In *Applied Cryptography and Network Security*, volume 3089 of *LNCS*, pages 46–60. Springer, 2004.

[69] Jens Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *ASIACRYPT'02: 12th International Conference on the Theory and Application of Cryptology and Information Security*, volume 4284 of *LNCS*, pages 444–459. Springer, 2006.

[70] Stuart Haber, Josh Benaloh, and Shai Halevi. The Helios e-Voting Demo for the IACR. International Association for Cryptologic Research. http://www.iacr.org/elections/eVoting/heliosDemo.pdf (accessed 7 May 2014), May 2010.

[71] Carmit Hazay and Yehuda Lindell. Sigma protocols and efficient zero-knowledge. In *Efficient Secure Two-Party Protocols*, Information Security and Cryptography, chapter 6, pages 147–175. Springer, 2010.

[72] James Heather and David Lundin. The Append-Only Web Bulletin Board. In *FAST'08: 5th International Workshop on Formal Aspects in Security and Trust*, volume 5491 of *LNCS*, pages 242–256. Springer, 2008.

[73] Martin Hirt. Receipt-Free K-out-of-L Voting Based on ElGamal Encryption. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 64–82. Springer, 2010.

[74] Martin Hirt and Kazue Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In *EUROCRYPT'06: 25th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 1807 of *LNCS*, pages 539–556. Springer, 2000.

[75] Benjamin Hosp and Poorvi L. Vora. An information-theoretic model of voting systems. In *WOTE'06: Workshop on Trustworthy Elections*, 2006.

[76] Benjamin Hosp and Poorvi L. Vora. An information-theoretic model of voting systems. *Mathematical and Computer Modelling*, 48(9-10):1628–1645, 2008.

[77] IACR Elections. http://www.iacr.org/elections/ (accessed 7 May 2014), 2013.

[78] Markus Jakobsson and Ari Juels. Mix and Match: Secure Function Evaluation via Ciphertexts. In *ASIACRYPT'00: 6th International Conference on the Theory and Application of Cryptology and Information Security*, volume 1976 of *LNCS*, pages 162–177. Springer, 2000.

[79] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. In *11th USENIX Security Symposium*, pages 339–353, 2002.

[80] Douglas W. Jones and Barbara Simons. *Broken Ballots: Will Your Vote Count?*, volume 204 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford University, 2012.

[81] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165, 2002.

[82] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *WPES'05: 4th Workshop on Privacy in the Electronic Society*, pages 61–70. ACM Press, 2005. See also http://www.colombia.rsa.com/rsalabs/staff/bios/ajuels/publications/Coercion/Coercion.pdf (accessed 7 May 2014).

[83] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 37–63. Springer, 2010.

[84] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[85] Shahram Khazaei and Douglas Wikström. Randomized partial checking revisited. In *CT-RSA'13: The Cryptographers' Track at the RSA Conference*, volume 7779 of *LNCS*, pages 115–128. Springer, 2013.

[86] Aggelos Kiayias. Electronic voting. In *Handbook of Financial Cryptography and Security*, chapter 3. Chapman and Hall/CRC, 2010.

[87] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT'15: 34th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 9057 of *LNCS*, pages 468–498. Springer, 2015.

[88] Steve Kremer and Mark D. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In *ESOP'05: 14th European Symposium on Programming*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

[89] Steve Kremer, Mark D. Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *ESORICS'10: 15th European Symposium on Research in Computer Security*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010.

[90] Ralf Kuesters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. Cryptology ePrint Archive, Report 2010/236 (version 20150202:163211), 2015.

[91] Ralf Küsters and Tomasz Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *S&P'09: 30th IEEE Symposium on Security and Privacy*, pages 251–266. IEEE Computer Society, 2009.

[92] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. In *CCS'10: 17th ACM Conference on Computer and Communications Security*, pages 526–535. ACM Press, 2010.

[93] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, Privacy, and Coercion-Resistance: New Insights from a Case Study. In *S&P'11: 32nd IEEE Symposium on Security and Privacy*, pages 538–553. IEEE Computer Society, 2011. Full version available at http://infsec.uni-trier.de/publications/paper/KuestersTruderungVogt-TR-2011.pdf.

[94] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A Game-Based Definition of Coercion-Resistance and its Applications. *Journal of Computer Security*, 20(6):709–764, 2012.

[95] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Clash Attacks on the Verifiability of E-Voting Systems. In *S&P'12: 33rd IEEE Symposium on Security and Privacy*, pages 395–409. IEEE Computer Society, 2012.

[96] Philip MacKenzie, Thomas Shrimpton, and Markus Jakobsson. Threshold password-authenticated key exchange. In *CRYPTO'02: 22nd International Cryptology Conference*, volume 2442 of *LNCS*, pages 385–400. Springer, 2002.

[97] Adam McCarthy, Ben Smyth, and Elizabeth A. Quaglia. Hawk and Aucitas: e-auction schemes from the Helios and Civitas e-voting schemes. In *FC'14: 18th International Conference on Financial Cryptography and Data Security*, volume 8437 of *LNCS*, pages 51–63. Springer, 2014.

[98] Tal Moran and Moni Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In *CRYPTO'06: 26th International Cryptology Conference*, volume 4117 of *LNCS*, pages 373–392. Springer, 2006.

[99] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *CCS'01: 8th ACM Conference on Computer and Communications Security*, pages 116–125. ACM Press, 2001.

[100] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Unpublished manuscript, 2004.

[101] NIST. Secure Hash Standard (SHS). FIPS PUB 180-4, Information Technology Laboratory, National Institute of Standards and Technology, March 2012.

[102] Helios Princeton Elections. https://princeton.heliosvoting.org/ (accessed 7 May 2014), 2012.

[103] Miriam Paiola and Bruno Blanchet. Verification of Security Protocols with Lists: From Length One to Unbounded Length. In *POST'12: First Conference on Principles of Security and Trust*, volume 7215 of *LNCS*, pages 69–88. Springer, 2012.

[104] Participants of the Dagstuhl Conference on Frontiers of E-Voting. *Dagstuhl Accord*, 2007. http://www.dagstuhlaccord.org/ (accessed 7 May 2014).

[105] R. A. Peters. A secure bulletin board. Master's thesis, Technische Universiteit Eindhoven, June 2005.

[106] Elizabeth A. Quaglia and Ben Smyth. Constructing secret, verifiable auction schemes from election schemes. Cryptology ePrint Archive, Report 2015/1204, 2015.

[107] Kazue Sako and Joe Kilian. Secure Voting Using Partially Compatible Homomorphisms. In *CRYPTO'94: 14th International Cryptology Conference*, volume 839 of *LNCS*, pages 411–424. Springer, 1994.

[108] Daniel Sandler and Dan S. Wallach. Casting votes in the Auditorium. In *EVT'07: Electronic Voting Technology Workshop*, Berkeley, CA, USA, 2007. USENIX Association.

[109] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89: 9th International Cryptology Conference*, volume 435 of *LNCS*, pages 239–252. Springer, 1990.

[110] Nicole Schweikardt. Arithmetic, first-order logic, and counting quantifiers. *ACM Transactions on Computational Logic*, 6(3):634–671, July 2005.

[111] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning*. PhD thesis, School of Computer Science, University of Birmingham, 2011.

[112] Ben Smyth. Replay attacks that violate ballot secrecy in helios. Cryptology ePrint Archive, Report 2012/185, 2012.

[113] Ben Smyth. Secrecy and independence for election schemes. Cryptology ePrint Archive, Report 2015/942, 2015.

[114] Ben Smyth, Myrto Arapinis, and Mark D Ryan. Translating between equational theories for automated reasoning. FCS'13: Workshop on Foundations of Computer Security, 2013.

[115] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In *ESORICS'13: 18th European Symposium on Research in Computer Security*, volume 8134 of *LNCS*, pages 463–480. Springer, 2013.

[116] Ben Smyth and David Bernhard. Ballot secrecy and ballot independence: definitions and relations. Cryptology ePrint Archive, Report 2013/235, 2014.

[117] Ben Smyth and David Bernhard. Ballot secrecy with malicious bulletin boards. Technical Report 2014/822, Cryptology ePrint Archive, 2014.

[118] Ben Smyth and Véronique Cortier. A note on replay attacks that violate privacy in electronic voting schemes. Technical Report RR-7643, INRIA, June 2011. http://hal.inria.fr/inria-00599182/.

[119] Ben Smyth, Yoshikazu Hanatani, and Hirofumi Muratani. NM-CPA secure encryption with proofs of plaintext knowledge. In *IWSEC'15: 10th International Workshop on Security*, volume 9241 of *LNCS*. Springer, 2015.

[120] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In *ARSPA-WITS'10: Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, volume 6186 of *LNCS*, pages 165–182. Springer, 2010.

[121] *Key issues and conclusions: May 2007 electoral pilot schemes*, May 2007. http://www.electoralcommission.org.uk/__data/assets/electoral_commission_pdf_file/0015/13218/Keyfindingsandrecommendationssummarypaper_27191-20111__E__N__S__W__.pdf (accessed 7 May 2014).

[122] Filip Zagórski, Richard T. Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *Applied Cryptography and Network Security*, volume 7954 of *LNCS*, pages 441–457. Springer, 2013.